

# Package ‘psborrow2’

May 14, 2026

**Type** Package

**Title** Bayesian Dynamic Borrowing Analysis and Simulation

**Version** 0.0.5.1

**Description** Bayesian dynamic borrowing is an approach to incorporating external data to supplement a randomized, controlled trial analysis in which external data are incorporated in a dynamic way (e.g., based on similarity of outcomes); see Viele 2013 <[doi:10.1002/pst.1589](https://doi.org/10.1002/pst.1589)> for an overview. This package implements the hierarchical commensurate prior approach to dynamic borrowing as described in Hobbes 2011 <[doi:10.1111/j.1541-0420.2011.01564.x](https://doi.org/10.1111/j.1541-0420.2011.01564.x)>. There are three main functionalities. First, 'psborrow2' provides a user-friendly interface for applying dynamic borrowing on the study results handles the Markov Chain Monte Carlo sampling on behalf of the user. Second, 'psborrow2' provides a simulation framework to compare different borrowing parameters (e.g. full borrowing, no borrowing, dynamic borrowing) and other trial and borrowing characteristics (e.g. sample size, covariates) in a unified way. Third, 'psborrow2' provides a set of functions to generate data for simulation studies, and also allows the user to specify their own data generation process. This package is designed to use the sampling functions from 'cmdstanr' which can be installed from <<https://stan-dev.r-universe.dev>>.

**URL** <https://github.com/Genentech/psborrow2>,  
<https://genentech.github.io/psborrow2/index.html>

**BugReports** <https://github.com/Genentech/psborrow2/issues>

**License** Apache License 2.0

**Encoding** UTF-8

**Depends** R (>= 4.1.0)

**RoxygenNote** 7.3.3

**Config/testthat/edition** 3

**Imports** checkmate, glue, methods, graphics, posterior, generics,  
Matrix, mvtnorm, future, simsurv

**Suggests** cmdstanr, survival, flexsurv, eha, testthat (>= 3.0), usethis (>= 2.1.5), vdiff, tibble, xml2, knitr, rmarkdown, bayesplot, matrixcalc, WeightIt, MatchIt, BayesPPD, ggsurvfit, gbm, ggplot2, cobalt, table1, gt, gtsummary

**Additional\_repositories** <https://stan-dev.r-universe.dev>

**Language** en-US

**SystemRequirements** cmdstan

**Collate** 'generics.R' 'prior\_class.R' 'covariate\_class.R'  
 'add\_covariates.R' 'prior\_normal.R' 'treatment\_class.R'  
 'borrowing\_class.R' 'outcome\_class.R' 'analysis\_class.R'  
 'borrowing\_details.R' 'borrowing\_fixed\_power\_prior.R'  
 'borrowing\_full.R' 'borrowing\_hierarchical\_commensurate.R'  
 'borrowing\_none.R' 'cast\_mat\_to\_long\_pem.R'  
 'check\_data\_matrix\_has\_columns.R' 'cmdstan.R'  
 'create\_analysis\_obj.R' 'create\_data\_matrix.R'  
 'treatment\_details.R' 'sim\_treatment\_list.R' 'helpers.R'  
 'outcome\_bin\_logistic.R' 'prior\_exponential.R'  
 'outcome\_surv\_weibull\_ph.R' 'outcome\_surv\_exponential.R'  
 'sim\_outcome\_list.R' 'sim\_borrowing\_list.R'  
 'sim\_covariate\_list.R' 'sim\_data\_list.R' 'simulation\_class.R'  
 'create\_simulation\_obj.R' 'data.R'  
 'get\_prior\_string\_covariates.R' 'prior\_half\_cauchy.R'  
 'outcome\_cont\_normal.R' 'outcome\_surv\_pem.R'  
 'load\_and\_interpolate\_stan\_model.R' 'load\_stan\_file.R'  
 'make\_analysis\_object\_list.R' 'mcmc\_sample.R'  
 'mcmc\_simulation\_result.R' 'package.R'  
 'prepare\_stan\_data\_inputs.R' 'prior\_bernoulli.R' 'prior\_beta.R'  
 'prior\_cauchy.R' 'prior\_gamma.R' 'prior\_half\_normal.R'  
 'prior\_poisson.R' 'sim\_covariates.R' 'sim\_estimate\_bias.R'  
 'sim\_estimate\_effect\_variance.R' 'sim\_estimate\_mse.R'  
 'sim\_is\_null\_effect\_covered.R' 'sim\_is\_true\_effect\_covered.R'  
 'sim\_samplesize.R' 'simulate\_data\_baseline.R' 'simulate\_data.R'  
 'simvar\_class.R' 'trim\_data\_matrix.R' 'uniform\_prior.R' 'zzz.R'

**LazyData** true

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Matt Secrest [aut, cre] (ORCID:  
 <<https://orcid.org/0000-0002-0939-4902>>),  
 Isaac Gravestock [aut],  
 Craig Gower-Page [ctb],  
 Manoj Khanal [ctb],  
 Mingyang Shan [ctb],  
 Kexin Jin [ctb],  
 Zhi Yang [ctb],  
 Genentech, Inc. [cph, fnd]

**Maintainer** Matt Secrest <[secrestm@gene.com](mailto:secrestm@gene.com)>

**Repository** CRAN

**Date/Publication** 2026-05-14 14:10:01 UTC

## Contents

add_covariates . . . . .	6
Analysis-class . . . . .	6
as_data_frame . . . . .	7
BaselineDataFrame-class . . . . .	8
BaselineDataList-class . . . . .	8
BaselineObject-class . . . . .	9
baseline_covariates . . . . .	9
bernoulli_prior . . . . .	10
beta_prior . . . . .	10
BinaryOutcome-class . . . . .	11
binary_cutoff . . . . .	11
bin_var . . . . .	12
Borrowing-class . . . . .	13
BorrowingFixedPowerPrior-class . . . . .	14
BorrowingFull-class . . . . .	14
BorrowingHierarchicalCommensurate-class . . . . .	15
BorrowingNone-class . . . . .	15
borrowing_details . . . . .	16
borrowing_fixed_power_prior . . . . .	16
borrowing_full . . . . .	17
borrowing_hierarchical_commensurate . . . . .	18
borrowing_none . . . . .	19
build_model_string . . . . .	20
c . . . . .	21
cauchy_prior . . . . .	21
check_cmdstanr . . . . .	22
check_data_matrix_has_columns . . . . .	22
check_fixed_external_data . . . . .	23
ContinuousOutcome-class . . . . .	24
cont_var . . . . .	24
covariance_matrix . . . . .	25
Covariates-class . . . . .	26
create_alpha_string . . . . .	26
create_analysis_obj . . . . .	27
create_baseline_object . . . . .	28
create_data_matrix . . . . .	29
create_data_simulation . . . . .	30
create_event_dist . . . . .	31
create_simulation_obj . . . . .	33
create_tau_string . . . . .	34
custom_enrollment . . . . .	35
cut_off_funs . . . . .	36
DataSimCutOff-class . . . . .	37
DataSimEnrollment-class . . . . .	37
DataSimEvent-class . . . . .	37
DataSimFixedExternalData-class . . . . .	38

DataSimObject-class . . . . .	38
enrollment_constant . . . . .	39
eval_constraints . . . . .	39
example_matrix . . . . .	40
example_surv . . . . .	40
exponential_prior . . . . .	41
exp_surv_dist . . . . .	42
gamma_prior . . . . .	42
generate . . . . .	43
generate,BaselineObject-method . . . . .	43
generate,DataSimObject-method . . . . .	44
get_cmd_stan_models . . . . .	45
get_data . . . . .	45
get_prior_string_covariates . . . . .	46
get_quantiles . . . . .	46
get_results . . . . .	47
get_stan_code . . . . .	47
get_vars . . . . .	48
half_cauchy_prior . . . . .	49
half_normal_prior . . . . .	50
load_and_interpolate_stan_model . . . . .	50
load_stan_file . . . . .	51
logistic_bin_outcome . . . . .	51
MCMCSimulationResult-class . . . . .	52
mcmc_sample . . . . .	52
normal_prior . . . . .	55
Outcome-class . . . . .	56
OutcomeBinaryLogistic-class . . . . .	56
OutcomeContinuousNormal-class . . . . .	57
OutcomeSurvExponential-class . . . . .	57
OutcomeSurvPEM-class . . . . .	58
OutcomeSurvWeibullPH-class . . . . .	59
outcome_bin_logistic . . . . .	59
outcome_cont_normal . . . . .	60
outcome_surv_exponential . . . . .	61
outcome_surv_pem . . . . .	63
outcome_surv_weibull_ph . . . . .	64
plot . . . . .	65
plot_pdf . . . . .	67
plot_pmf . . . . .	68
poisson_prior . . . . .	68
possible_data_sim_vars . . . . .	69
Prior-class . . . . .	69
PriorBernoulli-class . . . . .	70
PriorBeta-class . . . . .	70
PriorCauchy-class . . . . .	71
PriorExponential-class . . . . .	71
PriorGamma-class . . . . .	72

PriorHalfCauchy-class . . . . .	72
PriorHalfNormal-class . . . . .	73
PriorNormal-class . . . . .	73
PriorPoisson-class . . . . .	74
prior_bernoulli . . . . .	74
prior_beta . . . . .	75
prior_cauchy . . . . .	76
prior_exponential . . . . .	76
prior_gamma . . . . .	77
prior_half_cauchy . . . . .	78
prior_half_normal . . . . .	78
prior_normal . . . . .	79
prior_poisson . . . . .	80
rename_draws_covariates . . . . .	81
set_cut_off . . . . .	82
set_dropout . . . . .	82
set_enrollment . . . . .	83
set_transformations . . . . .	84
set_transformations,BaselineObject-method . . . . .	85
show_guide . . . . .	86
SimBorrowingList-class . . . . .	86
SimCovariateList-class . . . . .	86
SimCovariates-class . . . . .	87
SimDataList-class . . . . .	87
SimOutcomeList-class . . . . .	88
SimSampleSize-class . . . . .	88
SimTreatmentList-class . . . . .	88
Simulation-class . . . . .	89
SimVar-class . . . . .	89
SimVarBin-class . . . . .	89
SimVarCont-class . . . . .	90
sim_borrowing_list . . . . .	91
sim_covariates . . . . .	92
sim_covariates_summ . . . . .	93
sim_covariate_list . . . . .	93
sim_data_list . . . . .	94
sim_outcome_list . . . . .	96
sim_samplesize . . . . .	97
sim_treatment_list . . . . .	98
TimeToEvent-class . . . . .	98
Treatment-class . . . . .	99
treatment_details . . . . .	100
trim_cols . . . . .	100
trim_rows . . . . .	101
UniformPrior-class . . . . .	102
uniform_prior . . . . .	102
variable_dictionary . . . . .	103
weib_ph_surv_dist . . . . .	103

---

add_covariates	<i>Add Covariates for Model Adjustment</i>
----------------	--

---

**Description**

Specify column names for adjustment variables in model matrix and prior distributions for the model parameters for these covariates (i.e., betas)

**Usage**

```
add_covariates(covariates, priors)
```

**Arguments**

covariates	character. Names of columns in the data matrix containing covariates to be adjusted for in the outcome model. Note: the external and treatment flags should not go here.
priors	Either a single object of class <code>Prior</code> specifying the prior distribution to apply to all covariates or a named list of distributions of class <code>Prior</code> , one for each covariate

**Value**

Object of class `Covariates`.

**Examples**

```
add_covariates(  
  covariates = c("a", "b"),  
  priors = list(  
    "a" = prior_normal(0, 1),  
    "b" = prior_normal(0, 2)  
  )  
)
```

---

Analysis-class	<i>Analysis Class</i>
----------------	-----------------------

---

**Description**

A class for defining Analysis details. Objects of class `Analysis` should not be created directly but by the constructor `create_analysis_obj()`.

**Slots**

`data_matrix` `matrix`. The data matrix, including all covariates to be adjusted for, all relevant outcome variables, and treatment arm and external control arm flags.

`covariates` `Covariate`. Object of class `Covariate` as output by the function `covariate_details()`.

`outcome` `Outcome`. Object of class `Outcome` as output by `outcome_surv_exponential()`, `outcome_surv_weibull_ph()`, or `outcome_bin_logistic()`.

`borrowing` `Borrowing`. Object of class `Borrowing` as output by `borrowing_full()`, `borrowing_none()`, or `borrowing_hierarchical_commensurate()`.

`treatment` `Treatment`. Object of class `Treatment` as output by `treatment_details()`.

`model_string` `character`. The string that contains the full Stan model code to be compiled.

`model` `CmdStanModel`. The compiled Stan model as output by `cmdstanr::cmdstan_model()`

`ready_to_sample` `logical`. Is the object ready to sample?

---

as\_data\_frame

*Coerce a psborrow2 object to a data frame*


---

**Description**

Creates `data.frame` objects from various classes in `psborrow2`

**Usage**

```
## S3 method for class 'BaselineDataList'
as.data.frame(x, ...)
```

**Arguments**

`x` object of type: [BaselineDataList](#)

`...` Optional arguments for passed to [data.frame](#)

**Value**

A `data.frame`

---

BaselineDataFrame-class

*Baseline Data Frame Object*

---

**Description**

Contains a generated baseline dataset for a single arm.

**Value**

A BaselineDataFrame

**Slots**

cov\_names character contains the names of covariates generated from the multivariate normal distribution

means numeric contains the means of generating distribution for the covariates in cov\_names

variances numeric contains the marginal variances of generating distribution for the covariates in cov\_names.

---

BaselineDataList-class

*Baseline Data Frame List*

---

**Description**

A named list of BaselineDataFrames with generated data for internal\_treated/internal\_control/external\_control groups

**Value**

A BaselineDataList

**Slots**

baseline\_object Simulated covariates definitions as BaselineObject. See [create\\_baseline\\_object\(\)](#)

---

BaselineObject-class    *BaselineObject class for data simulation*

---

### Description

BaselineObject class for data simulation

### Slots

n\_trt\_int integer. Number of internal treated patients  
n\_ctrl\_int integer. Number of internal control patients  
n\_ctrl\_ext integer. Number of external control patients  
covariates list. List of correlated covariates objects, see [baseline\\_covariates\(\)](#)  
transformations list. List of named transformation functions.

---

baseline\_covariates    *Specify Correlated Baseline Covariates*

---

### Description

Set parameters to generate correlated multivariate normal data for internal and external patients.

### Usage

```
baseline_covariates(
  names,
  means_int,
  means_ext = means_int,
  covariance_int,
  covariance_ext = covariance_int
)
```

### Arguments

names                    character vector of variable names.  
means\_int                numeric vector of means for internal patients. Must have same length as names  
means\_ext                numeric vector of means for external patients. Must have same length as names  
covariance\_int          variance-covariance matrix for generating multivariate normal for internal patients. Must be square matrix with same number of rows and length(names)  
covariance\_ext          variance-covariance matrix for generating multivariate normal data for external patients. Must be square matrix with same number of rows and length(names)

**Value**

[BaselineObject](#) to build simulated dataset

**Examples**

```
corr_covs <- baseline_covariates(  
  names = c("b1", "b2"),  
  means_int = c(5, 25),  
  covariance_int = covariance_matrix(diag = c(1, 1), upper_tri = 0.4)  
)
```

---

bernoulli\_prior      *Legacy function for the bernoulli prior*

---

**Description**

Please use `prior_bernoulli()` instead.

**Usage**

```
bernoulli_prior(...)
```

**Arguments**

...      Deprecated arguments to `bernoulli_prior()`.

**Value**

This function does not return a value. When called, it triggers an error message indicating that `bernoulli_prior()` is deprecated and that `prior_bernoulli()` should be used instead.

---

beta\_prior      *Legacy function for the beta prior*

---

**Description**

Please use `prior_beta()` instead.

**Usage**

```
beta_prior(...)
```

**Arguments**

...      Deprecated arguments to `beta_prior()`.

**Value**

This function does not return a value. When called, it triggers an error message indicating that `beta_prior()` is deprecated and that `prior_beta()` should be used instead.

---

BinaryOutcome-class    BinaryOutcome *class*

---

**Description**

BinaryOutcome class

**Slots**

`n_param` integer. Number of ancillary parameters for the model to estimate.  
`param_priors` list. Named list of prior distributions on the ancillary parameters in the model.  
`binary_var` character. Variable used for outcome in BinaryOutcome objects.  
`baseline_prior` Prior. Object of class Prior specifying prior distribution for the baseline outcome.  
`name_beta_trt`. Named vector for `beta_trt`.  
`name_exp_trt`. Named vector for exponentiated `beta_trt`  
`alpha_type`. How to interpret alpha.  
`name_addnl_params`. Named vector for additional parameters.

**See Also**

Other outcome: [ContinuousOutcome-class](#), [Outcome-class](#), [OutcomeBinaryLogistic-class](#), [OutcomeContinuousNormal-class](#), [OutcomeSurvExponential-class](#), [OutcomeSurvPEM-class](#), [OutcomeSurvWeibullPH-class](#), [TimeToEvent-class](#)

---

binary\_cutoff                      *Binary Cut-Off Transformation*

---

**Description**

Binary Cut-Off Transformation

**Usage**

`binary_cutoff(name, int_cutoff, ext_cutoff)`

**Arguments**

name	variable to transform
int_cutoff	cut-off for internal patients, numeric between 0 and 1
ext_cutoff	cut-off for external patients, numeric between 0 and 1

**Value**

Transformation function to be used in `create_baseline_object()`. Sets quantile values larger than cut-off value to TRUE otherwise FALSE.

**Examples**

```
# Creates a simple function, where `data` is a `BaselineDataFrame`:
function(data) {
  ext <- data$ext == 0
  q <- get_quantiles(data, name)
  ifelse(ext, q > int_cutoff, q > ext_cutoff)
}
```

---

bin\_var

*Create binary covariate*


---

**Description**

Create an object of class `SimVarBin` to hold proportions of binary variables specified in a simulation study.

**Usage**

```
bin_var(
  prob_internal,
  prob_external,
  mu_internal_before_bin = 0,
  mu_external_before_bin = 0
)
```

**Arguments**

prob_internal	numeric. Proportion for the internal arms.
prob_external	numeric. Proportion for the external arm.
mu_internal_before_bin	numeric. Mean value of the covariate before binarization for the internal arms. The default is 0. See details for more information.
mu_external_before_bin	numeric. Mean value of the covariate before binarization for the external arm. The default is 0. See details for more information.

**Details**

This function contains information necessary to create binary covariates as part of a simulation study. The binary covariates are created by binarizing multivariate normal distributions to achieve the probabilities specified in `prob_internal` and `prob_external`. The user may choose to change the default mean value of each variable prior to binarization by specifying `mu_internal_before_bin` or `mu_external_before_bin` to ensure the correct scales are used in the covariance matrix, though the ultimate proportions will depend on `prob_internal` and `prob_external`. The default values for `mu_internal_before_bin` and `mu_external_before_bin` are 0, and it is not recommended to change these without good reason.

**Value**

Object of class `SimVarBin`.

**See Also**

Other simvar: `cont_var()`

**Examples**

```
cv1 <- bin_var(0.50, 0.80)
cv2 <- bin_var(.95, .92)
```

---

Borrowing-class

Borrowing Class

---

**Description**

A class for defining borrowing details. Objects of class `Borrowing` should not be created directly but by the constructors `borrowing_hierarchical_commensurate()`, `borrowing_none()`, `borrowing_full()`.

**Slots**

`method_name` string. The name of the method.

`ext_flag_col` character. Name of the external flag column in the matrix.

**See Also**

Prior constructor functions: `borrowing_full()`, `borrowing_hierarchical_commensurate()`, `borrowing_none()`

Other borrowing classes: `BorrowingFixedPowerPrior-class`, `BorrowingFull-class`, `BorrowingHierarchicalCommensurate-class`, `BorrowingNone-class`

---

 BorrowingFixedPowerPrior-class

 BorrowingFixedPowerPrior *class*


---

**Description**

BorrowingFixedPowerPrior class

**Slots**

method\_name string. The name of the method.

ext\_flag\_col character. Name of the external flag column in the matrix.

powers numeric. The fixed values to be used as the powers in the power prior.

**See Also**

Other borrowing classes: [Borrowing-class](#), [BorrowingFull-class](#), [BorrowingHierarchicalCommensurate-class](#), [BorrowingNone-class](#)

---

 BorrowingFull-class

 BorrowingFull *class*


---

**Description**

A class for defining details for "Full Borrowing" methods. Objects of class BorrowingFull should not be created directly but by the constructor [borrowing\\_full\(\)](#).

**Slots**

method\_name string. The name of the method.

ext\_flag\_col character. Name of the external flag column in the matrix.

name\_tau named vector for hierarchical commensurability parameter hyperprior.

**See Also**

Other borrowing classes: [Borrowing-class](#), [BorrowingFixedPowerPrior-class](#), [BorrowingHierarchicalCommensurate-class](#), [BorrowingNone-class](#)

---

BorrowingHierarchicalCommensurate-class  
 BorrowingHierarchicalCommensurate *class*

---

**Description**

A class for defining details of dynamic borrowing using the hierarchical Bayesian model with a commensurability parameter. Objects of class BorrowingHierarchicalCommensurate should not be created directly but by the constructor [borrowing\\_hierarchical\\_commensurate\(\)](#).

**Slots**

`method_name` string. The name of the method.  
`ext_flag_col` character. Name of the external flag column in the matrix.  
`tau_prior` Prior. Prior for the commensurability parameter.

**See Also**

Other borrowing classes: [Borrowing-class](#), [BorrowingFixedPowerPrior-class](#), [BorrowingFull-class](#), [BorrowingNone-class](#)

---

BorrowingNone-class    BorrowingNone *class*

---

**Description**

A class for defining details for "No borrowing" methods. Objects of class BorrowingNone should not be created directly but by the constructor [borrowing\\_none\(\)](#).

**Slots**

`method_name` string. The name of the method.  
`ext_flag_col` character. Name of the external flag column in the matrix.

**See Also**

Other borrowing classes: [Borrowing-class](#), [BorrowingFixedPowerPrior-class](#), [BorrowingFull-class](#), [BorrowingHierarchicalCommensurate-class](#)

---

borrowing_details	<i>Legacy function for specifying borrowing details</i>
-------------------	---

---

**Description**

Please use one of `borrowing_hierarchical_commensurate()`, `borrowing_none()`, or `borrowing_full()` instead.

**Usage**

```
borrowing_details(...)
```

**Arguments**

...           Deprecated arguments to `borrowing_details`.

**Value**

This function does not return a value. When called, it triggers an error message indicating that `borrowing_details()` is deprecated and that one of `borrowing_hierarchical_commensurate()`, `borrowing_none()`, or `borrowing_full()` should be used instead.

---

borrowing_fixed_power_prior	<i>Fixed Power Prior Borrowing</i>
-----------------------------	------------------------------------

---

**Description**

Fixed Power Prior Borrowing

**Usage**

```
borrowing_fixed_power_prior(ext_flag_col, power_par)
```

**Arguments**

`ext_flag_col`   character. Name of the external flag column in the matrix.  
`power_par`       numeric. Fixed power parameter for all external data. Must be between 0 and 1.

**Value**

Object of class `BorrowingFixedPowerPrior`.

**Examples**

```
borrowing_fixed_power_prior(  
  ext_flag_col = "ext",  
  power_par = 0.5  
)
```

---

borrowing_full	<i>Full borrowing</i>
----------------	-----------------------

---

**Description**

Full borrowing

**Usage**

```
borrowing_full(ext_flag_col)
```

**Arguments**

ext\_flag\_col    character. Name of the external flag column in the matrix.

**Details****Method:**

This method does not distinguish between internal and external arms, effectively pooling patients.

**External Control:**

The ext\_flag\_col argument refers to the column in the data matrix that contains the flag indicating a patient is from the external control cohort.

**Value**

Object of class [BorrowingFull](#).

**See Also**

Other borrowing: [borrowing\\_none\(\)](#)

**Examples**

```
fb <- borrowing_full("ext")
```

---

borrowing\_hierarchical\_commensurate

*Hierarchical commensurate borrowing*

---

## Description

Hierarchical commensurate borrowing

## Usage

```
borrowing_hierarchical_commensurate(ext_flag_col, tau_prior)
```

## Arguments

`ext_flag_col` character. Name of the external flag column in the matrix.  
`tau_prior` Prior. Prior for the commensurability parameter.

## Details

### Method:

In Bayesian dynamic borrowing using the hierarchical commensurate prior approach, external control information is borrowed to the extent that the outcomes (i.e., log hazard rates or log odds) are similar between external and internal control populations. See Viele 2014 [doi:10.1002/pst.1589](https://doi.org/10.1002/pst.1589) and Hobbs 2011 [doi:10.1111/j.15410420.2011.01564.x](https://doi.org/10.1111/j.15410420.2011.01564.x) for details.

### External Control:

The `ext_flag_col` argument refers to the column in the data matrix that contains the flag indicating a patient is from the external control cohort.

### Tau Prior:

The `tau_prior` argument specifies the hyperprior on the precision parameter commonly referred to as the commensurability parameter. See Viele 2014 [doi:10.1002/pst.1589](https://doi.org/10.1002/pst.1589) for more details. This hyperprior determines (along with the comparability of the outcomes between internal and external controls) how much borrowing of the external control group will be performed. Example hyperpriors include largely uninformative inverse gamma distributions [e.g., `prior_gamma(alpha = .001, beta = .001)`] as well as more informative distributions [e.g., `prior_gamma(alpha = 1, beta = .001)`], though any distribution  $x \in (0, \infty)$  can be used. Distributions with more density at higher values of  $x$  (i.e., higher precision) will lead to more borrowing.

## Value

Object of class `BorrowingHierarchicalCommensurate`.

## References

Viele, K., Berry, S., Neuenschwander, B., Amzal, B., Chen, F., Enas, N., Hobbs, B., Ibrahim, J.G., Kinnersley, N., Lindborg, S., Micallef, S., Roychoudhury, S. and Thompson, L. (2014), Use of historical control data for assessing treatment effects in clinical trials. **Pharmaceut. Statist.**, **13**: 41–54. doi:10.1002/pst.1589

Hobbes, B.P., Carlin, B.P., Mandrekar, S.J. and Sargent, D.J. (2011), Hierarchical commensurate and power prior models for adaptive incorporation of historical information in clinical trials. **Biometrics**, **67**: 1047–1056. doi:10.1111/j.15410420.2011.01564.x

## Examples

```
db <- borrowing_hierarchical_commensurate(
  ext_flag_col = "ext",
  tau_prior = prior_gamma(0.0001, 0.0001)
)
```

---

borrowing_none	<i>No borrowing</i>
----------------	---------------------

---

## Description

No borrowing

## Usage

```
borrowing_none(ext_flag_col)
```

## Arguments

`ext_flag_col` character. Name of the external flag column in the matrix.

## Details

### Method:

This method evaluates only the internal comparison, ignoring historical controls. Note that this method will filter the model matrix based on values in `ext_flag_col`.

### External Control:

The `ext_flag_col` argument refers to the column in the data matrix that contains the flag indicating a patient is from the external control cohort.

## Value

Object of class `BorrowingNone`.

## See Also

Other borrowing: `borrowing_full()`

## Examples

```
db <- borrowing_none(  
  ext_flag_col = "ext"  
)
```

---

build\_model\_string      *Build the model string by interpolating the Stan template*

---

## Description

Build the model string by interpolating the Stan template

## Usage

```
build_model_string(  
  template_domain,  
  template_filename,  
  outcome,  
  borrowing,  
  analysis_obj,  
  ...  
)
```

## Arguments

template_domain	Character string specifying the domain of the template (e.g., "surv", "bin", "cont")
template_filename	Character string specifying the filename of the Stan template
outcome	Outcome object
borrowing	Borrowing object
analysis_obj	Analysis object
...	Additional named arguments to be passed for interpolation

## Value

String containing the interpolated Stan model

---

c	<i>Combine objects in psborrow2</i>
---	-------------------------------------

---

**Description**

Creates `data.frame` objects from various classes in `psborrow2`

**Usage**

```
## S4 method for signature 'SimDataList'  
c(x, ...)
```

**Arguments**

x	object of type: <a href="#">SimDataList</a>
...	additional objects to combine

**Value**

A combined object  
list of [SimDataList](#) objects.

---

cauchy_prior	<i>Legacy function for the cauchy prior</i>
--------------	---

---

**Description**

Please use `prior_cauchy()` instead.

**Usage**

```
cauchy_prior(...)
```

**Arguments**

...	Deprecated arguments to <code>cauchy_prior()</code> .
-----	---

**Value**

This function does not return a value. When called, it triggers an error message indicating that `cauchy_prior()` is deprecated and that `prior_cauchy()` should be used instead.

---

check_cmdstanr	<i>Check Stan</i>
----------------	-------------------

---

**Description**

Check whether cmdstanr is available and prints version and logistic example.

**Usage**

```
check_cmdstanr(check_sampling = FALSE)
```

```
check_cmdstanr()
```

**Arguments**

`check_sampling` Compile and sample from the "logistic" example model.

**Value**

`check_cmdstanr()` prints results from checks.

`check_cmdstanr()` returns TRUE if CmdStan seems to be installed, otherwise FALSE

**Functions**

- `check_cmdstanr()`: Check if the CmdStan command line tools are available.

**Examples**

```
check_cmdstanr()
```

---

check_data_matrix_has_columns	<i>Check Data Matrix for Required Columns</i>
-------------------------------	---

---

**Description**

Check that an Analysis object's `data_matrix` has all the required variables.

**Usage**

```
check_data_matrix_has_columns(object)
```

**Arguments**

`object` Analysis. Object to check.

**Value**

stop() if some columns are missing.

**Examples**

```

anls <- create_analysis_obj(
  data_matrix = example_matrix,
  covariates = add_covariates(
    covariates = c("cov1", "cov2"),
    priors = prior_normal(0, 1000)
  ),
  outcome = outcome_surv_exponential(
    "time",
    "cnsr",
    baseline_prior = prior_normal(0, 1000)
  ),
  borrowing = borrowing_hierarchical_commensurate(
    "ext",
    prior_exponential(.001)
  ),
  treatment = treatment_details(
    "trt",
    prior_normal(0, 1000)
  )
)

check_data_matrix_has_columns(anls)

```

---

check\_fixed\_external\_data

*Create a Fixed External Data Object*

---

**Description**

Create a Fixed External Data Object

**Usage**

```
check_fixed_external_data(data, req_cols)
```

**Arguments**

data	A data.frame containing external control data
req_cols	A character vector of required covariate columns

**Value**

A DataSimObject with updated enrollment\_internal and enrollment\_external slots.

---

ContinuousOutcome-class  
ContinuousOutcome *class*

---

### Description

ContinuousOutcome class

### Slots

n\_param integer. Number of ancillary parameters for the model to estimate.  
 param\_priors list. Named list of prior distributions on the ancillary parameters in the model.  
 continuous\_var character. Variable used for outcome in ContinuousOutcome objects.  
 baseline\_prior Prior. Object of class Prior specifying prior distribution for the baseline outcome.  
 name\_beta\_trt. Named vector for beta\_trt.  
 alpha\_type. How to interpret alpha.  
 name\_addnl\_params. Named vector for additional parameters.

### See Also

Other outcome: [BinaryOutcome-class](#), [Outcome-class](#), [OutcomeBinaryLogistic-class](#), [OutcomeContinuousNormal-class](#), [OutcomeSurvExponential-class](#), [OutcomeSurvPEM-class](#), [OutcomeSurvWeibullPH-class](#), [TimeToEvent-class](#)

---

cont\_var                      *Create continuous covariate*

---

### Description

Create an object of class SimVarCont to hold mean values of of continuous variables specified in a simulation study.

### Usage

```
cont_var(mu_internal, mu_external)
```

### Arguments

mu\_internal      numeric. Mean covariate value for the internal arms.  
 mu\_external      numeric. Mean covariate value for the external arm.

### Value

Object of class [SimVarCont](#).

**See Also**

Other simvar: [bin\\_var\(\)](#)

**Examples**

```
cv1 <- cont_var(0.5, 1)
cv2 <- cont_var(10, 10)
```

---

covariance\_matrix      *Create Covariance Matrix*

---

**Description**

Create Covariance Matrix

**Usage**

```
covariance_matrix(diag, upper_tri)
```

**Arguments**

diag                  Diagonal entries of the covariance matrix  
upper\_tri            Upper triangle entries of the matrix, specified column wise.

**Value**

A symmetric matrix with `diag` values on the main diagonal and `upper_tri` values in the lower and upper triangles.

**Examples**

```
m1 <- covariance_matrix(c(1, 1, 1, 1), c(.8, .3, .8, 0, 0, 0))
m1
mvtnorm::rmvnorm(5, mean = c(0, 0, 0, 0), sigma = m1)

# No correlation
covariance_matrix(c(1, 2, 3))
```

---

Covariates-class	Covariate <i>Class</i>
------------------	------------------------

---

**Description**

A class for defining covariate details. Objects of class `Covariate` should not be created directly but by the constructor `add_covariates()`.

**Slots**

`covariates` character. Names of columns in the data matrix containing covariates to be adjusted for in the outcome model. Note: the external and treatment flags should not go here.

`priors`. Either a single object of class `Prior` specifying the prior distribution to apply to all covariates or a named list of distributions of class `Prior`, one for each covariate

`name_betas`. Names for the beta parameters in the STAN model.

---

create_alpha_string	<i>Create alpha string</i>
---------------------	----------------------------

---

**Description**

Create alpha string

**Usage**

```
create_alpha_string(borrowing_object, outcome_object)
```

```
## S4 method for signature 'Borrowing'
create_alpha_string(borrowing_object, outcome_object)
```

```
## S4 method for signature 'BorrowingHierarchicalCommensurate'
create_alpha_string(borrowing_object, outcome_object)
```

**Arguments**

`borrowing_object`  
                   borrowing object

`outcome_object` outcome object

---

create\_analysis\_obj     *Compile MCMC sampler using STAN and create analysis object*

---

## Description

Compile MCMC sampler using STAN and create analysis object

## Usage

```
create_analysis_obj(  
  data_matrix,  
  outcome,  
  borrowing,  
  treatment,  
  covariates = NULL,  
  quiet = FALSE  
)
```

## Arguments

data_matrix	matrix. The data matrix, including all covariates to be adjusted for, all relevant outcome variables, and treatment arm and external control arm flags.
outcome	Outcome. Object of class <a href="#">Outcome</a> as output by <a href="#">outcome_surv_exponential()</a> , <a href="#">outcome_surv_weibull_ph()</a> , or <a href="#">outcome_bin_logistic()</a> .
borrowing	Borrowing. Object of class <a href="#">Borrowing</a> as output by <a href="#">borrowing_full()</a> , <a href="#">borrowing_none()</a> , and <a href="#">borrowing_hierarchical_commensurate()</a> .
treatment	Treatment. Object of class <a href="#">Treatment</a> as output by <a href="#">treatment_details()</a> .
covariates	Covariates. Object of class <a href="#">Covariates</a> as output by the function <a href="#">add_covariates()</a> .
quiet	logical. Whether to suppress messages (TRUE) or not (FALSE, the default)

## Value

Object of class [Analysis](#).

## Examples

```
if (check_cmdstan()) {  
  anls <- create_analysis_obj(  
    data_matrix = example_matrix,  
    outcome = outcome_surv_exponential(  
      "time",  
      "cnsr",  
      baseline_prior = prior_normal(0, 1000)  
    ),  
    borrowing = borrowing_hierarchical_commensurate(  
      "ext",  
      prior_exponential(.001)  
    )  
  )  
}
```

```

    ),
    treatment = treatment_details(
      "trt",
      prior_normal(0, 1000)
    ),
    covariates = add_covariates(
      covariates = c("cov1", "cov2"),
      priors = prior_normal(0, 1000)
    )
  )
}

```

---

create\_baseline\_object

*Create Baseline Data Simulation Object*

---

## Description

Create Baseline Data Simulation Object

## Usage

```

create_baseline_object(
  n_trt_int,
  n_ctrl_int,
  n_ctrl_ext,
  covariates,
  transformations
)

```

## Arguments

n_trt_int	Number of internal treated patients
n_ctrl_int	Number of internal control patients
n_ctrl_ext	Number of external control patients
covariates	List of correlated covariates objects, see <a href="#">baseline_covariates()</a>
transformations	List of named transformation functions.

## Details

Transformation functions are evaluated in order and create or overwrite a column in the `data.frame` with that name. The function should take a `data.frame` (specifically a `BaselineDataFrame` object from `generate(BaselineObject)`) and return a vector with length identical to the total number of patients. The `@BaselineObject` slot may be accessed directly or with `get_quantiles()` to create transformations. See [binary\\_cutoff\(\)](#)

**Value**

A [BaselineObject](#)

**Examples**

```
bl_no_covs <- create_baseline_object(  
  n_trt_int = 100,  
  n_ctrl_int = 50,  
  n_ctrl_ext = 100  
)  
  
bl_biomarkers <- create_baseline_object(  
  n_trt_int = 100,  
  n_ctrl_int = 50,  
  n_ctrl_ext = 100,  
  covariates = baseline_covariates(  
    c("b1", "b2", "b3"),  
    means_int = c(0, 0, 0),  
    covariance_int = covariance_matrix(c(1, 1, 1), c(.8, .3, .8))  
  ),  
  transformations = list(  
    exp_b1 = function(data) exp(data$b1),  
    b2 = binary_cutoff("b2", int_cutoff = 0.7, ext_cutoff = 0.5)  
  )  
)
```

---

create\_data\_matrix      *Create Data Matrix*

---

**Description**

Creates a matrix suitable for [create\\_analysis\\_obj\(\)](#). Creates dummy variables for factors and allows transformations of covariates specified with a formula.

**Usage**

```
create_data_matrix(  
  data,  
  outcome,  
  trt_flag_col,  
  ext_flag_col,  
  covariates = NULL,  
  weight_var = NULL  
)
```

**Arguments**

data	data.frame. Data containing all variables
outcome	character. The outcome variable for binary outcomes or the time and censoring variables.
trt_flag_col	character. The treatment indicator variable.
ext_flag_col	character. The external cohort indicator.
covariates	character or formula. The covariates for model adjustment.
weight_var	character. An optional weight variable.

**Value**

Invisibly returns a matrix containing all variables to pass to `create_analysis_obj()`. Prints names of covariates columns to use with `add_covariates()`.

**Examples**

```
dat <- survival::diabetic
dat$ext <- dat$trt == 0 & dat$id > 1000
data_mat <- create_data_matrix(
  dat,
  outcome = c("time", "status"),
  trt_flag_col = "trt",
  ext_flag_col = "ext",
  covariates = ~ age + laser + log(risk)
)
data_mat
```

---

```
create_data_simulation
```

*Data Simulation*

---

**Description**

Data Simulation

**Usage**

```
create_data_simulation(
  baseline,
  coefficients = numeric(),
  treatment_hr = 1,
  drift_hr = 1,
  event_dist,
  fixed_external_data
)
```

**Arguments**

baseline	BaselineObject from <a href="#">create_baseline_object()</a>
coefficients	Named vector of coefficients for linear predictor. Must correspond to variables in baseline object
treatment_hr	Default treatment hazard ratio for simulations. Alternative simulation settings can be specified in <a href="#">generate</a> .
drift_hr	Default drift hazard ratio between internal and external arms. Alternative simulation settings can be specified in <a href="#">generate</a> .
event_dist	Specify time to event distribution with SimDataEvent object from <a href="#">create_event_dist()</a>
fixed_external_data	A data.frame containing external control data. It must contain columns eventtime, status and all of the variables named in coefficients. If present, trt must be 0 and ext must be 1 for all rows.

**Value**

DataSimObject

**Examples**

```
baseline_obj <- create_baseline_object(
  n_trt_int = 100,
  n_ctrl_int = 50,
  n_ctrl_ext = 10,
  covariates = baseline_covariates(
    names = c("age", "score"),
    means_int = c(55, 5),
    means_ext = c(60, 5),
    covariance_int = covariance_matrix(c(5, 1))
  )
)
sim_obj <- create_data_simulation(
  baseline_obj,
  coefficients = c(age = 0.001, score = 1.5),
  event_dist = create_event_dist(dist = "exponential", lambdas = 1 / 36)
)
data_sim_list <- generate(sim_obj, treatment_hr = c(0.5, 1), drift_hr = 0.5)
```

---

create\_event\_dist      *Specify a Time to Event Distribution*

---

**Description**

Uses [simsurv::simsurv](#) to generate time to event data. See [simsurv](#) help for more details.

**Usage**

```

create_event_dist(
  dist = NULL,
  lambdas = NULL,
  gammas = NULL,
  mixture = FALSE,
  pmix = 0.5,
  hazard = NULL,
  loghazard = NULL,
  cumhazard = NULL,
  logcumhazard = NULL,
  ...
)

null_event_dist()

```

**Arguments**

dist	Specify the distribution "exponential"
lambdas	Scale parameter
gammas	Second parameter needed for Weibull or Gompertz distributions
mixture	Use mixture model?
pmix	Proportion of mixtures
hazard	A user defined hazard function
loghazard	Alternatively, a user defined log hazard function
cumhazard	Alternatively, a user defined cumulative hazard function
logcumhazard	Alternatively, a user defined log cumulative hazard function
...	Other simsurv parameters

**Value**

A SimDataEvent object

null\_event\_dist returns an object with no parameters specified that does not simulate event times.

**Examples**

```

weibull_surv <- create_event_dist(dist = "weibull", lambdas = 1 / 200, gammas = 1)
exp_event_dist <- create_event_dist(dist = "exponential", lambdas = 1 / 36)
null_event_dist()

```

---

create\_simulation\_obj *Compile MCMC sampler using STAN and create simulation object*

---

## Description

Compile MCMC sampler using STAN and create simulation object

## Usage

```
create_simulation_obj(
  data_matrix_list,
  covariate = NULL,
  outcome,
  borrowing,
  treatment,
  quiet = TRUE
)
```

## Arguments

data_matrix_list	SimDataList. The list of lists of data matrices created with <code>sim_data_list()</code> .
covariate	SimCovariateList or Covariate or NULL. List of Covariate objects created with <code>sim_covariate()</code> , a single Covariate object created by <code>add_covariates()</code> , or NULL (no covariate adjustment).
outcome	SimOutcomeList or Outcome. List of Outcome objects created with <code>sim_outcome()</code> , or single Outcome object (e.g., created by <code>outcome_surv_exponential()</code> ).
borrowing	SimBorrowingList or Borrowing. List of Borrowing objects created with <code>sim_borrowing()</code> , or a single Borrowing object created by <code>borrowing_full()</code> , <code>borrowing_none()</code> , or <code>borrowing_hierarchical_commensurate()</code> .
treatment	SimTreatmentList or Treatment. List of Treatment objects created with <code>sim_treatment()</code> or a single Treatment object created by <code>treatment_details()</code> .
quiet	logical. Whether to print messages ( <code>quiet = FALSE</code> ) or not ( <code>quiet = TRUE</code> , the default)

## Value

Object of class [Simulation](#).

## Examples

```
base_mat <- matrix(
  c(
    rep(0, 200), rep(0, 200), rep(1, 200),
    rep(1, 200), rep(0, 200), rep(0, 200),
    rep(0, 600)
  )
```

```

),
ncol = 3,
dimnames = list(NULL, c("ext", "trt", "driftOR"))
)

add_binary_endpoint <- function(odds_ratio,
                                base_matrix = base_mat) {
  linear_predictor <- base_matrix[, "trt"] * log(odds_ratio)
  prob <- 1 / (1 + exp(-linear_predictor))

  bin_endpoint <- rbinom(
    NROW(base_matrix),
    1,
    prob
  )

  cbind(base_matrix, matrix(bin_endpoint, ncol = 1, dimnames = list(NULL, "ep")))
}

data_list <- list(
  list(add_binary_endpoint(1.5), add_binary_endpoint(1.5)),
  list(add_binary_endpoint(2.5), add_binary_endpoint(2.5))
)

guide <- data.frame(
  trueOR = c(1.5, 2.5),
  driftOR = c(1.0, 1.0),
  index = 1:2
)

sdl <- sim_data_list(
  data_list = data_list,
  guide = guide,
  effect = "trueOR",
  drift = "driftOR",
  index = "index"
)

if (check_cmdstan()) {
  sim_object <- create_simulation_obj(
    data_matrix_list = sdl,
    outcome = outcome_bin_logistic("ep", prior_normal(0, 1000)),
    borrowing = sim_borrowing_list(list(
      full_borrowing = borrowing_full("ext"),
      bdb = borrowing_hierarchical_commensurate("ext", prior_exponential(0.0001))
    )),
    treatment = treatment_details("trt", prior_normal(0, 1000))
  )
}

```

**Description**

Create tau string

**Usage**

```
create_tau_string(borrowing_object)

## S4 method for signature 'Borrowing'
create_tau_string(borrowing_object)

## S4 method for signature 'BorrowingHierarchicalCommensurate'
create_tau_string(borrowing_object)
```

**Arguments**

borrowing\_object  
                   borrowing object

---

custom\_enrollment      *Create a DataSimEnrollment Object*

---

**Description**

Create a DataSimEnrollment Object

**Usage**

```
custom_enrollment(fun, label)
```

**Arguments**

fun                   A function that takes one argument n the number of enrollment times to observe and returns a vector of times.

label                 A user-friendly label

**Value**

A [DataSimEnrollment](#) object

**Examples**

```
custom_enrollment(
  fun = function(n) rpois(n, lambda = 5),
  label = "Poisson enrollment distribution"
)
```

---

`cut_off_funs`*Cut Off Functions*

---

**Description**

Cut Off Functions

**Usage**

```
cut_off_none()
cut_off_after_first(time)
cut_off_after_last(time)
cut_off_after_events(n)
```

**Arguments**

<code>time</code>	Time to cut off
<code>n</code>	Number of events

**Value**

A `DataSimCutOff` object containing a cut-off function

**Functions**

- `cut_off_none()`: No cut off is specified
- `cut_off_after_first()`: Cut off at time after first enrolled patient
- `cut_off_after_last()`: Cut off at time after last enrolled patient
- `cut_off_after_events()`: Cut off after the time of the n-th event

**Examples**

```
cut_off_none()
cut_off_after_first(time = 36)
cut_off_after_last(time = 36)
cut_off_after_events(n = 20)
```



---

DataSimFixedExternalData-class

*Fixed External Control Data Object*

---

### Description

Fixed External Control Data Object

### Value

A FixedExternalData

### Slots

data `data.frame` containing external control data

n Number of observations

---

DataSimObject-class

*Data Simulation Object Class*

---

### Description

Data Simulation Object Class

### Value

A DataSimObject

### Slots

baseline BaselineObject from [create\\_baseline\\_object](#)

coefficients Named numeric vector of beta coefficients for survival model. See beta at `?simsurv::simsurv`

treatment\_hr numeric treatment effect as a hazard ration. `log(treatment_hr)` is included in beta with coefficients and `log(drift_hr)`. This default is overridden by [generate](#) arguments

drift\_hr numeric hazard ratio between internal and external arms. Included as `log(drift_hr)`.

fixed\_external\_data `data.frame` for external data. Currently unused.

event\_dist DataSimEvent parameters for outcome distribution from [create\\_event\\_dist\(\)](#)

enrollment DataSimEnrollment object.

cut\_off DataSimCutOff

---

enrollment\_constant     *Constant Enrollment Rates*

---

**Description**

Constant Enrollment Rates

**Usage**

```
enrollment_constant(rate, for_time = rep(1, length(rate)))
```

**Arguments**

rate                    Number of patients to enroll per unit time  
for\_time                Number of time periods for each rate. Must be equal length to rate

**Value**

An object of class [DataSimEnrollment](#) to be passed to [create\\_data\\_simulation\(\)](#)

**Examples**

```
# 10 patients/month for 6 months, then 5/month for 6 months  
enroll_obj <- enrollment_constant(rate = c(10, 5), for_time = c(6, 6))  
enroll_obj@fun(n = 80)
```

---

eval\_constraints        *Evaluate constraints*

---

**Description**

Evaluate constraints when these are called

**Usage**

```
eval_constraints(object)  
  
## S4 method for signature 'Prior'  
eval_constraints(object)
```

**Arguments**

object                Prior object

---

example_matrix	<i>Example data matrix</i>
----------------	----------------------------

---

**Description**

A matrix containing data from a clinical trial with a treatment arm and a control arm, as well as data from an external control. In this simulated dataset, the true hazard ratio (HR) for the time-to-event endpoint comparing the experimental treatment to the control treatment is 0.70. The true odds ratio (OR) for the binary response endpoint comparing the experimental treatment to the control treatment is 1.20.

**Usage**

```
example_matrix
```

**Format**

A data frame with 500 rows and 11 columns. The distributions of patients is: 50 internal control patients, 100 internal experimental patients, 350 external control patients.

**id** patient identifier

**ext** 0/1, flag for external controls

**trt** 0/1, flag for treatment arm

**cov1** 0/1, baseline covariate

**cov2** 0/1, baseline covariate

**cov3** 0/1, baseline covariate

**cov4** 0/1, baseline covariate

**time** numeric >0, survival time

**status** 0/1, indicator for event status (1 = had event, 0 = did not have event)

**cnsr** 0/1, censoring indicator (1 = was censored, 0 = was not censored). This value is 1 - status.

**resp** 0/1, indicator for response outcome (1 = had a response, 0 = did not have a response)

---

example_surv	<i>Simulated Survival Data</i>
--------------	--------------------------------

---

**Description**

A data frame containing simulated data from a clinical trial with a treatment arm (n=200) and a control arm (n=158), as well as data from an external control (n=242).

**Usage**

```
example_surv
```

**Format**

A data frame with 600 rows and 6 variables:

**trt** 0/1, flag for treatment arm

**ext** 0/1, flag for external controls

**eventtime** numeric >0, survival time

**status** 0/1, event indicator

**ensor** 0/1, censoring indicator

**cov1** 0/1, binary baseline covariate 1

**cov2** integer in [0, 15], baseline covariate 2

**cov3** continuous numeric, baseline covariate 3

---

exponential\_prior      *Legacy function for the exponential prior*

---

**Description**

Please use `prior_exponential()` instead.

**Usage**

```
exponential_prior(...)
```

**Arguments**

...                      Deprecated arguments to `exponential_prior()`.

**Value**

This function does not return a value. When called, it triggers an error message indicating that `exponential_prior()` is deprecated and that `prior_exponential()` should be used instead.

---

exp_surv_dist	<i>Legacy function for the exponential survival distribution</i>
---------------	--

---

**Description**

Please use `outcome_surv_exponential()` instead.

**Usage**

```
exp_surv_dist(...)
```

**Arguments**

...           Deprecated arguments to `exp_surv_dist()`.

**Value**

This function does not return a value. When called, it triggers an error message indicating that `exp_surv_dist()` is deprecated and that `outcome_surv_exponential()` should be used instead.

---

gamma_prior	<i>Legacy function for the gamma prior</i>
-------------	--

---

**Description**

Please use `prior_gamma()` instead.

**Usage**

```
gamma_prior(...)
```

**Arguments**

...           Deprecated arguments to `gamma_prior()`.

**Value**

This function does not return a value. When called, it triggers an error message indicating that `gamma_prior()` is deprecated and that `prior_gamma()` should be used instead.

---

generate	<i>Generate Data from Object</i>
----------	----------------------------------

---

**Description**

Generate Data from Object

**Usage**

```
generate(x, ...)
```

**Arguments**

x	object
...	Other arguments passed to methods

**Value**

Object of class [SimDataList](#).

---

generate, BaselineObject-method
<i>Generate Data for a BaselineObject</i>

---

**Description**

Generate Data for a BaselineObject

**Usage**

```
## S4 method for signature 'BaselineObject'  
generate(x, ...)
```

**Arguments**

x	a BaselineObject object created by <a href="#">create_baseline_object</a>
...	additional parameters are ignored

**Value**

A [BaselineDataFrame](#) object

**Examples**

```

bl_biomarkers <- create_baseline_object(
  n_trt_int = 100,
  n_ctrl_int = 50,
  n_ctrl_ext = 100,
  covariates = baseline_covariates(
    c("b1", "b2", "b3"),
    means_int = c(0, 0, 0),
    covariance_int = covariance_matrix(c(1, 1, 1), c(.8, .3, .8))
  ),
  transformations = list(
    exp_b1 = function(data) exp(data$b1),
    b2 = binary_cutoff("b2", int_cutoff = 0.7, ext_cutoff = 0.5)
  )
)
generate(bl_biomarkers)

```

---

generate,DataSimObject-method

*Generate Data for a DataSimObject*

---

**Description**

Generate Data for a DataSimObject

**Usage**

```

## S4 method for signature 'DataSimObject'
generate(x, n = 1, treatment_hr = NULL, drift_hr = NULL)

```

**Arguments**

x	a DataSimObject object created by <a href="#">create_data_simulation</a>
n	number of data sets to simulate
treatment_hr	vector of numeric treatment effects
drift_hr	vector of numeric drift effects

**Value**

A [SimDataList](#) object for use with [create\\_simulation\\_obj\(\)](#).

**Examples**

```

baseline_obj <- create_baseline_object(
  n_trt_int = 100,
  n_ctrl_int = 50,
  n_ctrl_ext = 10,
  covariates = baseline_covariates(

```

```

    names = c("age", "score"),
    means_int = c(55, 5),
    means_ext = c(60, 5),
    covariance_int = covariance_matrix(c(5, 1))
  )
)
sim_obj <- create_data_simulation(
  baseline_obj,
  coefficients = c(age = 0.001, score = 1.5),
  event_dist = create_event_dist(dist = "exponential", lambdas = 1 / 36)
)
data_sim_list <- generate(sim_obj, treatment_hr = c(0, 1), drift_hr = 0.5)

```

---

```
get_cmd_stan_models    Get CmdStanModel objects for MCMCSimulationResults
```

---

### Description

Show the CmdStanModel objects from MCMCSimulationResults objects.

### Usage

```

get_cmd_stan_models(object)

## S4 method for signature 'MCMCSimulationResult'
get_cmd_stan_models(object)

```

### Arguments

object                    MCMCSimulationResults object

### Value

List of lists of CmdStanModel objects for each model.

---

```
get_data                Get Simulated Data from SimDataList object
```

---

### Description

Retrieves the simulated data from a SimDataList object by index.

### Usage

```

get_data(object, index = 1, dataset = 1)

## S4 method for signature 'SimDataList'
get_data(object, index = NULL, dataset = NULL)

```

**Arguments**

object	SimDataList object
index	the index of the scenario (see guide with print(SimDataList))
dataset	the dataset out of n_datasets_per_param

**Value**

Simulated data as a data frame if the index is specified, else as a list

---

get\_prior\_string\_covariates  
*Get prior string for all covariates*

---

**Description**

Get prior string for all covariates

**Usage**

```
get_prior_string_covariates(covariates)
```

**Arguments**

covariates	Covariates object
------------	-------------------

---

get\_quantiles *Get Quantiles of Random Data*

---

**Description**

Helper for use within transformation functions for [create\\_baseline\\_object\(\)](#).

**Usage**

```
get_quantiles(object, var)
```

**Arguments**

object	a BaselineDataFrame
var	character string name of the variable

**Value**

A numeric vector containing quantiles based on the data generating distribution.

---

get_results	<i>Get results for MCMCSimulationResults objects</i>
-------------	--

---

**Description**

Get the results data.frame from MCMCSimulationResults objects.

**Usage**

```
get_results(object)

## S4 method for signature 'MCMCSimulationResult'
get_results(object)
```

**Arguments**

object            MCMCSimulationResults object

**Value**

data.frame with simulation results.

---

get_stan_code	<i>Get method for Stan model</i>
---------------	----------------------------------

---

**Description**

Get method for Stan model

**Usage**

```
get_stan_code(object)

## S4 method for signature 'Analysis'
get_stan_code(object)
```

**Arguments**

object            Analysis object

**Value**

String containing the Stan model

---

`get_vars`*Get Variables*

---

**Description**

Gets the data variable names from an object.

**Usage**

```
get_vars(object)

## S4 method for signature 'Covariates'
get_vars(object)

## S4 method for signature 'Treatment'
get_vars(object)

## S4 method for signature 'Borrowing'
get_vars(object)

## S4 method for signature 'TimeToEvent'
get_vars(object)

## S4 method for signature 'BinaryOutcome'
get_vars(object)

## S4 method for signature 'ContinuousOutcome'
get_vars(object)

## S4 method for signature 'Analysis'
get_vars(object)

## S4 method for signature 'NULL'
get_vars(object)

## S4 method for signature 'BorrowingFixedPowerPrior'
get_vars(object)

## S4 method for signature 'SimTreatmentList'
get_vars(object)

## S4 method for signature 'SimOutcomeList'
get_vars(object)

## S4 method for signature 'SimBorrowingList'
get_vars(object)
```

```
## S4 method for signature 'SimCovariateList'  
get_vars(object)  
  
## S4 method for signature 'Simulation'  
get_vars(object)  
  
## S4 method for signature 'BaselineObject'  
get_vars(object)
```

### Arguments

object            Object

### Value

A character vector containing variable names

### Examples

```
get_vars(treatment_details(  
  trt_flag_col = "treat_fl",  
  trt_prior = prior_normal(0, 1000)  
))
```

---

half\_cauchy\_prior        *Legacy function for the half-cauchy prior*

---

### Description

Please use `prior_half_cauchy()` instead.

### Usage

```
half_cauchy_prior(...)
```

### Arguments

...                Deprecated arguments to `half_cauchy_prior()`.

### Value

This function does not return a value. When called, it triggers an error message indicating that `half_cauchy_prior()` is deprecated and that `prior_half_cauchy()` should be used instead.

---

half_normal_prior	<i>Legacy function for the normal half prior</i>
-------------------	--

---

**Description**

Please use `prior_half_normal()` instead.

**Usage**

```
half_normal_prior(...)
```

**Arguments**

...           Deprecated arguments to `half_normal_prior()`.

**Value**

This function does not return a value. When called, it triggers an error message indicating that `half_normal_prior()` is deprecated and that `prior_half_normal()` should be used instead.

---

load_and_interpolate_stan_model	<i>Load and interpolate Stan model</i>
---------------------------------	--

---

**Description**

Load and interpolate Stan model

**Usage**

```
load_and_interpolate_stan_model(outcome, borrowing, analysis_obj)
```

**Arguments**

outcome	Outcome object
borrowing	Borrowing object
analysis_obj	Analysis object

**Value**

String containing the interpolated Stan model

---

load_stan_file	<i>Load a Stan psborrow2 template</i>
----------------	---------------------------------------

---

**Description**

This function loads a Stan template file from the package's 'inst/stan' directory.

**Usage**

```
load_stan_file(...)
```

**Arguments**

...                   subdirectories

**Value**

template string

---

logistic_bin_outcome	<i>Legacy function for binary logistic regression</i>
----------------------	---

---

**Description**

Please use `outcome_bin_logistic()` instead.

**Usage**

```
logistic_bin_outcome(...)
```

**Arguments**

...                   Deprecated arguments to `logistic_bin_outcome`.

**Value**

This function does not return a value. When called, it triggers an error message indicating that `logistic_bin_outcome()` is deprecated and that `outcome_bin_logistic()` should be used instead.

---

MCMCSimulationResult-class  
MCMCSimulationResult *Class*

---

### Description

A class for defining Simulation study results. Objects of class MCMCSimulationResult should not be created directly but by `mcmc_sample()`.

### Slots

`results` *data.frame*. The results of the simulation study summarized in a *data.frame*  
`cmd_stan_models` *list*. List of lists of *CmdStanmodels* corresponding to the different parameters in `Simulation@guide` and different datasets in `Simulation@data_matrix_list`.

---

`mcmc_sample`                      *Sample from Stan model*

---

### Description

Method to sample from compiled Stan model and return a *CmdStanMCMC* object with draws.

### Usage

```
mcmc_sample(x, ...)
```

```
## S4 method for signature 'ANY'
```

```
mcmc_sample(x, ...)
```

```
## S4 method for signature 'Analysis'
```

```
mcmc_sample(
  x,
  iter_warmup = 1000L,
  iter_sampling = 10000L,
  chains = 4L,
  verbose = FALSE,
  ...
)
```

```
## S4 method for signature 'Simulation'
```

```
mcmc_sample(
  x,
  posterior_quantiles = c(0.025, 0.975),
  iter_warmup = 1000L,
  iter_sampling = 10000L,
```

```

    chains = 4L,
    verbose = FALSE,
    keep_cmdstan_models = FALSE,
    ...
  )

```

## Arguments

x	object to sample, such as Analysis (created with <code>create_analysis_obj()</code> ) or Simulation.
...	additional arguments passed to the <code>\$sample()</code> method of a <code>cmdstanr</code> Stan model. See <a href="https://mc-stan.org/cmdstanr/reference/model-method-sample.html">https://mc-stan.org/cmdstanr/reference/model-method-sample.html</a>
iter_warmup	integer. The number of warm up iterations to run per chain. The default is 1000.
iter_sampling	integer. The number of post-warm up iterations to run per chain. The default is 10000.
chains	integer. The number of Markov chains to run. The default is 4.
verbose	logical. Whether to print sampler updates (TRUE) or not (FALSE)
posterior_quantiles	numeric vector of length two. The posterior quantiles used for summarizing simulation results. The default is <code>c(0.025, 0.975)</code> See details.
keep_cmdstan_models	logical. Whether to keep the <code>CmdStanModel</code> objects from the <code>mcmc_sampler</code> (TRUE, discouraged in most scenarios) or not (FALSE). The default is FALSE.

## Details

### Simulation objects:

This function takes draws from an MCMC sampler and summarizes results.

## Value

An object of class `CmdStanMCMC`

An object of class `MCMCSimulationResult`

## Examples

```

## Analysis objects
if (check_cmdstan()) {
  anls <- create_analysis_obj(
    data_matrix = example_matrix,
    covariates = add_covariates(
      covariates = c("cov1", "cov2"),
      priors = prior_normal(0, 1000)
    ),
    outcome = outcome_surv_weibull_ph(
      "time",
      "cnsr",
      shape_prior = prior_normal(0, 1000),

```

```

    baseline_prior = prior_normal(0, 1000)
  ),
  borrowing = borrowing_hierarchical_commensurate(
    "ext",
    prior_exponential(.001)
  ),
  treatment = treatment_details("trt", prior_normal(0, 1000))
)

mcmc_results <- mcmc_sample(anls, chains = 1, iter_warmup = 500L, iter_sampling = 1000L)
}

## Simulation objects
base_mat <- matrix(
  c(
    rep(0, 200), rep(0, 200), rep(1, 200),
    rep(1, 200), rep(0, 200), rep(0, 200),
    rep(0, 600)
  ),
  ncol = 3,
  dimnames = list(NULL, c("ext", "trt", "driftOR"))
)

add_binary_endpoint <- function(odds_ratio,
                                base_matrix = base_mat) {
  linear_predictor <- base_matrix[, "trt"] * log(odds_ratio)
  prob <- 1 / (1 + exp(-linear_predictor))

  bin_endpoint <- rbinom(
    NROW(base_matrix),
    1,
    prob
  )

  cbind(base_matrix, matrix(bin_endpoint, ncol = 1, dimnames = list(NULL, "ep")))
}

data_list <- list(
  list(add_binary_endpoint(1.5), add_binary_endpoint(1.5)),
  list(add_binary_endpoint(2.5), add_binary_endpoint(2.5))
)

guide <- data.frame(
  trueOR = c(1.5, 2.5),
  driftOR = c(1.0, 1.0),
  index = 1:2
)

sdl <- sim_data_list(
  data_list = data_list,
  guide = guide,
  effect = "trueOR",
  drift = "driftOR",

```

```

    index = "index"
  )

  if (check_cmdstan()) {
    sim_object <- create_simulation_obj(
      data_matrix_list = sdl,
      outcome = outcome_bin_logistic("ep", prior_normal(0, 1000)),
      borrowing = sim_borrowing_list(list(
        full_borrowing = borrowing_full("ext"),
        bdb = borrowing_hierarchical_commensurate("ext", prior_exponential(0.0001))
      )),
      treatment = treatment_details("trt", prior_normal(0, 1000))
    )

    mcmc_sample(sim_object, chains = 1, iter_warmup = 500L, iter_sampling = 1000L)
  }
  ## Not run:
  library(future)
  # Use two separate R processes
  plan("multisession", workers = 2)

  # and two parallel threads in each.
  mcmc_sample(sim_object, chains = 1, iter_warmup = 500L, iter_sampling = 1000L, parallel_chains = 2)

  # Tidy up processes when finished
  plan("sequential")

  ## End(Not run)

```

---

normal\_prior

*Legacy function for the normal prior*


---

### Description

Please use `prior_normal()` instead.

### Usage

```
normal_prior(...)
```

### Arguments

...                   Deprecated arguments to `normal_prior()`.

### Value

This function does not return a value. When called, it triggers an error message indicating that `normal_prior()` is deprecated and that `prior_normal()` should be used instead.

---

Outcome-class	Outcome <i>class</i>
---------------	----------------------

---

**Description**

Outcome class

**See Also**

Other outcome: [BinaryOutcome-class](#), [ContinuousOutcome-class](#), [OutcomeBinaryLogistic-class](#), [OutcomeContinuousNormal-class](#), [OutcomeSurvExponential-class](#), [OutcomeSurvPEM-class](#), [OutcomeSurvWeibullPH-class](#), [TimeToEvent-class](#)

---

OutcomeBinaryLogistic-class	OutcomeBinaryLogistic <i>class</i>
-----------------------------	------------------------------------

---

**Description**

A class for defining a logistic regression with a binary outcome to be translated to Stan code. Objects of class OutcomeBinaryLogistic should not be created directly but by the constructor [outcome\\_bin\\_logistic\(\)](#).

**Slots**

`n_param` integer. Number of ancillary parameters for the model to estimate (0).

`param_priors` list. Named list of prior distributions on the ancillary parameters in the model. Empty for OutcomeBinaryLogistic.

`binary_var` character. Variable used for outcome in OutcomeBinaryLogistic objects.

`baseline_prior` Prior. Object of class Prior specifying prior distribution for the baseline outcome.

`name_beta_trt`. Named vector for beta\_trt.

`name_exp_trt`. Named vector for exponentiated beta\_trt

`alpha_type`. How to interpret alpha.

`name_addnl_params`. Named vector for additional parameters.

**See Also**

Other outcome: [BinaryOutcome-class](#), [ContinuousOutcome-class](#), [Outcome-class](#), [OutcomeContinuousNormal-class](#), [OutcomeSurvExponential-class](#), [OutcomeSurvPEM-class](#), [OutcomeSurvWeibullPH-class](#), [TimeToEvent-class](#)

---

OutcomeContinuousNormal-class  
OutcomeContinuousNormal *class*

---

**Description**

A class for defining a regression with a normal outcome to be translated to Stan code. Objects of class OutcomeContinuousNormal should not be created directly but by the constructor [outcome\\_cont\\_normal\(\)](#).

**Slots**

`n_param` integer. Number of ancillary parameters for the model to estimate (0).  
`param_priors` list. Named list of prior distributions on the ancillary parameters in the model.  
 Empty for OutcomeContinuousNormal.  
`continuous_var` character. Variable used for outcome in OutcomeContinuousNormal objects.  
`baseline_prior` Prior. Object of class Prior specifying prior distribution for the baseline outcome.  
`name_beta_trt`. Named vector for beta\_trt.  
`alpha_type`. How to interpret alpha.  
`name_addnl_params`. Named vector for additional parameters.

**See Also**

Other outcome: [BinaryOutcome-class](#), [ContinuousOutcome-class](#), [Outcome-class](#), [OutcomeBinaryLogistic-class](#), [OutcomeSurvExponential-class](#), [OutcomeSurvPEM-class](#), [OutcomeSurvWeibullPH-class](#), [TimeToEvent-class](#)

---

OutcomeSurvExponential-class  
OutcomeSurvExponential *Class*

---

**Description**

A class for defining a time-to-event survival analysis with an exponential survival distribution. Objects of class OutcomeSurvExponential should not be created directly but by the constructor [outcome\\_surv\\_exponential\(\)](#).

**Slots**

`n_param` integer. Number of ancillary parameters for the model to estimate (0).  
`param_priors` list. Named list of prior distributions on the ancillary parameters in the model.  
 Empty for OutcomeSurvExponential.  
`time_var` character. Variable used for time in TimeToEvent objects.  
`cens_var` character. Variable used for censoring in TimeToEvent objects.

baseline\_prior Prior. Object of class Prior specifying prior distribution for the baseline outcome.

name\_beta\_trt. Named vector for beta\_trt.

name\_exp\_trt. Named vector for exponentiated beta\_trt

alpha\_type. How to interpret alpha.

name\_addnl\_params. Named vector for additional parameters.

### See Also

Other outcome: [BinaryOutcome-class](#), [ContinuousOutcome-class](#), [Outcome-class](#), [OutcomeBinaryLogistic-class](#), [OutcomeContinuousNormal-class](#), [OutcomeSurvPEM-class](#), [OutcomeSurvWeibullPH-class](#), [TimeToEvent-class](#)

---

OutcomeSurvPEM-class    OutcomeSurvPEM Class

---

### Description

A class for defining a time-to-event survival analysis with a piecewise survival distribution. Objects of class OutcomeSurvPEM should not be created directly but by the constructor [outcome\\_surv\\_pem\(\)](#).

### Slots

function\_stan\_code character. stan function code block containing text to interpolate into stan model. Empty string for OutcomeSurvPEM.

param\_stan\_code character. stan parameter code block containing text to interpolate into stan model. Empty string for OutcomeSurvPEM.

n\_param integer. Number of ancillary parameters for the model to estimate (0).

param\_priors list. Named list of prior distributions on the ancillary parameters in the model. Empty for OutcomeSurvPEM.

time\_var character. Variable used for time in TimeToEvent objects.

cens\_var character. Variable used for censoring in TimeToEvent objects.

baseline\_prior Prior. Object of class Prior specifying prior distribution for the baseline outcome.

name\_beta\_trt. Named vector for beta\_trt.

name\_exp\_trt. Named vector for exponentiated beta\_trt

alpha\_type. How to interpret alpha.

name\_addnl\_params. Named vector for additional parameters.

n\_periods. Number of periods.

### See Also

Other outcome: [BinaryOutcome-class](#), [ContinuousOutcome-class](#), [Outcome-class](#), [OutcomeBinaryLogistic-class](#), [OutcomeContinuousNormal-class](#), [OutcomeSurvExponential-class](#), [OutcomeSurvWeibullPH-class](#), [TimeToEvent-class](#)

---

 OutcomeSurvWeibullPH-class

 OutcomeSurvWeibullPH Class
 

---

### Description

A class for defining a time-to-event survival analysis with a Weibull proportional hazards survival distribution. Objects of class OutcomeSurvWeibullPH should not be created directly but by the constructor `outcome_surv_weibull_ph()`.

### Slots

`n_param` integer. Number of ancillary parameters for the model to estimate (1).

`param_priors` list. Named list of prior distributions on the ancillary parameters in the model.

`time_var` character. Variable used for time in TimeToEvent objects.

`cens_var` character. Variable used for censoring in TimeToEvent objects.

`baseline_prior` Prior. Object of class Prior specifying prior distribution for the baseline outcome.

`name_beta_trt`. Named vector for beta\_trt.

`name_exp_trt`. Named vector for exponentiated beta\_trt

`alpha_type`. How to interpret alpha.

`name_addnl_params`. Named vector for additional parameters.

### See Also

Other outcome: [BinaryOutcome-class](#), [ContinuousOutcome-class](#), [Outcome-class](#), [OutcomeBinaryLogistic-class](#), [OutcomeContinuousNormal-class](#), [OutcomeSurvExponential-class](#), [OutcomeSurvPEM-class](#), [TimeToEvent-class](#)

---

 outcome\_bin\_logistic *Bernoulli distribution with logit parametrization*


---

### Description

Bernoulli distribution with logit parametrization

### Usage

```
outcome_bin_logistic(binary_var, baseline_prior, weight_var = "")
```

**Arguments**

binary_var	character. Name of binary (1/0 or TRUE/FALSE) outcome variable in the model matrix
baseline_prior	Prior. Object of class Prior specifying prior distribution for the baseline outcome. See Details for more information.
weight_var	character. Optional name of variable in model matrix for weighting the log likelihood.

**Details****Baseline Prior:**

The baseline\_prior argument specifies the prior distribution for the baseline log odds. The interpretation of the baseline\_prior differs slightly between borrowing methods selected.

- *Dynamic borrowing using* borrowing\_hierarchical\_commensurate(): the baseline\_prior for Bayesian Dynamic Borrowing refers to the log odds of the external control arm.
- *Full borrowing or No borrowing using* borrowing\_full() or borrowing\_none(): the baseline\_prior for these borrowing methods refers to the log odds for the internal control arm.

**Value**

Object of class [OutcomeBinaryLogistic](#).

**See Also**

Other outcome models: [outcome\\_cont\\_normal\(\)](#), [outcome\\_surv\\_exponential\(\)](#), [outcome\\_surv\\_pem\(\)](#), [outcome\\_surv\\_weibull\\_ph\(\)](#)

**Examples**

```
lg <- outcome_bin_logistic(
  binary_var = "response",
  baseline_prior = prior_normal(0, 1000)
)
```

---

outcome\_cont\_normal    *Normal Outcome Distribution*

---

**Description**

Normal Outcome Distribution

**Usage**

```
outcome_cont_normal(
  continuous_var,
  baseline_prior,
  std_dev_prior,
  weight_var = ""
)
```

**Arguments**

continuous_var	character. Name of continuous outcome variable in the model matrix
baseline_prior	Prior. Object of class Prior specifying prior distribution for the baseline outcome. See Details for more information.
std_dev_prior	Prior. Object of class Prior specifying prior distribution for the standard deviation of the outcome distribution (i.e. "sigma").
weight_var	character. Optional name of variable in model matrix for weighting the log likelihood.

**Details****Baseline Prior:**

The `baseline_prior` argument specifies the prior distribution for the intercept of the linear model. The interpretation of the `baseline_prior` differs slightly between borrowing methods selected.

- *Dynamic borrowing using* `borrowing_hierarchical_commensurate()`: the `baseline_prior` for Bayesian Dynamic Borrowing refers to the intercept of the external control arm.
- *Full borrowing or No borrowing using* `borrowing_full()` or `borrowing_none()`: the `baseline_prior` for these borrowing methods refers to the intercept for the internal control arm.

**Value**

Object of class `OutcomeContinuousNormal`.

**See Also**

Other outcome models: `outcome_bin_logistic()`, `outcome_surv_exponential()`, `outcome_surv_pem()`, `outcome_surv_weibull_ph()`

**Examples**

```
norm <- outcome_cont_normal(
  continuous_var = "tumor_size",
  baseline_prior = prior_normal(0, 100),
  std_dev_prior = prior_half_cauchy(1, 5)
)
```

---

outcome\_surv\_exponential

*Exponential survival distribution*

---

**Description**

Exponential survival distribution

## Usage

```
outcome_surv_exponential(time_var, cens_var, baseline_prior, weight_var = "")
```

## Arguments

time_var	character. Name of time variable column in model matrix
cens_var	character. Name of the censorship variable flag in model matrix
baseline_prior	Prior. Object of class <code>Prior</code> specifying prior distribution for the baseline outcome. See <code>Details</code> for more information.
weight_var	character. Optional name of variable in model matrix for weighting the log likelihood.

## Details

### Baseline Prior:

The `baseline_prior` argument specifies the prior distribution for the baseline log hazard rate. The interpretation of the `baseline_prior` differs slightly between borrowing methods selected.

- *Dynamic borrowing using* `borrowing_hierarchical_commensurate()`: the `baseline_prior` for Bayesian Dynamic Borrowing refers to the log hazard rate of the external control arm.
- *Full borrowing or No borrowing using* `borrowing_full()` or `borrowing_none()`: the `baseline_prior` for these borrowing methods refers to the log hazard rate for the internal control arm.

## Value

Object of class `OutcomeSurvExponential`.

## See Also

Other outcome models: `outcome_bin_logistic()`, `outcome_cont_normal()`, `outcome_surv_pem()`, `outcome_surv_weibull_ph()`

## Examples

```
es <- outcome_surv_exponential(  
  time_var = "time",  
  cens_var = "cens",  
  baseline_prior = prior_normal(0, 1000)  
)
```

---

outcome_surv_pem	<i>Piecewise exponential survival distribution</i>
------------------	--

---

## Description

Piecewise exponential survival distribution

## Usage

```
outcome_surv_pem(
  time_var,
  cens_var,
  baseline_prior,
  weight_var = "",
  cut_points
)
```

## Arguments

time_var	character. Name of time variable column in model matrix
cens_var	character. Name of the censorship variable flag in model matrix
baseline_prior	Prior. Object of class Prior specifying prior distribution for each cut point. See Details for more information.
weight_var	character. Optional name of variable in model matrix for weighting the log likelihood.
cut_points	numeric. Vector of internal cut points for the piecewise exponential model. Note: the choice of cut points will impact the amount of borrowing between arms when dynamic borrowing methods are selected. It is recommended to choose cut points that contain an equal number of events within each interval. Please include only internal cut points in the vector. For instance, for cut points of [0, 15], (15, 20], (20, Inf), the vector should be c(15, 20). If you pass cut-points beyond the follow-up of the data, you will receive an informative warning when calling create_analysis_object() and these cut points will be ignored.

## Details

### Baseline Prior:

The baseline\_prior argument specifies the prior distribution for the baseline log hazard rate within each cutpoint. Currently, there is no option to consider different baseline priors within each cut point. The interpretation of the baseline\_prior differs slightly between borrowing methods selected.

- *Dynamic borrowing using* borrowing\_hierarchical\_commensurate(): the baseline\_prior for Bayesian Dynamic Borrowing refers to the log hazard rate of the external control arm.
- *Full borrowing or No borrowing using* borrowing\_full() or borrowing\_none(): the baseline\_prior for these borrowing methods refers to the log hazard rate for the internal control arm.

**Value**

Object of class `OutcomeSurvPEM`.

**See Also**

Other outcome models: `outcome_bin_logistic()`, `outcome_cont_normal()`, `outcome_surv_exponential()`, `outcome_surv_weibull_ph()`

**Examples**

```
es <- outcome_surv_pem(
  time_var = "time",
  cens_var = "cens",
  baseline_prior = prior_normal(0, 1000),
  cut_points = c(10, 15, 30)
)
```

---

`outcome_surv_weibull_ph`

*Weibull survival distribution (proportional hazards formulation)*

---

**Description**

Weibull survival distribution (proportional hazards formulation)

**Usage**

```
outcome_surv_weibull_ph(
  time_var,
  cens_var,
  shape_prior,
  baseline_prior,
  weight_var = ""
)
```

**Arguments**

<code>time_var</code>	character. Name of time variable column in model matrix
<code>cens_var</code>	character. Name of the censorship variable flag in model matrix
<code>shape_prior</code>	Prior class object for the Weibull shape parameter. Default is <code>prior_exponential(beta = 0.0001)</code> .
<code>baseline_prior</code>	Prior. Object of class <code>Prior</code> specifying prior distribution for the baseline outcome. See <code>Details</code> for more information.
<code>weight_var</code>	character. Optional name of variable in model matrix for weighting the log likelihood.

**Details****Baseline Prior:**

The `baseline_prior` argument specifies the prior distribution for the baseline log hazard rate. The interpretation of the `baseline_prior` differs slightly between borrowing methods selected.

- *Dynamic borrowing using* `borrowing_hierarchical_commensurate()`: the `baseline_prior` for Bayesian Dynamic Borrowing refers to the log hazard rate of the external control arm.
- *Full borrowing or No borrowing using* `borrowing_full()` or `borrowing_none()`: the `baseline_prior` for these borrowing methods refers to the log hazard rate for the internal control arm.

**Value**

Object of class `OutcomeSurvWeibullPH`.

**See Also**

Other outcome models: `outcome_bin_logistic()`, `outcome_cont_normal()`, `outcome_surv_exponential()`, `outcome_surv_pem()`

**Examples**

```
ws <- outcome_surv_weibull_ph(
  time_var = "time",
  cens_var = "cens",
  shape_prior = prior_exponential(1),
  baseline_prior = prior_normal(0, 1000)
)
```

---

 plot

*Plot Prior Objects*


---

**Description**

Plot prior distributions as densities. Continuous distributions are plotted as curves and discrete distributions as bar plots.

**Usage**

```
## S4 method for signature 'Prior,missing'
plot(
  x,
  y,
  default_limits,
  dist_type = c("continuous", "discrete"),
  density_fun,
  add,
  ...
)
```

```
## S4 method for signature 'PriorNormal,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'PriorExponential,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'PriorHalfCauchy,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'PriorBernoulli,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'PriorBeta,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'PriorCauchy,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'PriorGamma,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'PriorHalfNormal,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'PriorPoisson,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'UniformPrior,missing'
plot(x, y, add = FALSE, ...)
```

### Arguments

<code>x</code>	Object inheriting from <code>Prior</code>
<code>y</code>	Not used.
<code>default_limits</code>	Numeric range to plot distribution over.
<code>dist_type</code>	Plot a continuous or discrete distribution.
<code>density_fun</code>	Function which takes a vector of values and returns a vector of density values.
<code>add</code>	logical. Add density to existing plot.
<code>...</code>	Optional arguments for plotting.

### Details

Plot ranges are selected by default to show 99% of the density for unbounded distributions. The limits can be changed by specifying `xlim = c(lower, upper)`.

Colors, line types, and other typical `par()` parameters can be used.

**Value**

No return value, this function generates a plot in the current graphics device.

**Examples**

```
plot(prior_normal(1, 2))
plot(prior_exponential(0.1))
plot(prior_half_cauchy(0, 1), xlim = c(-20, 20))
plot(prior_half_cauchy(0, 2), xlim = c(-20, 20), col = 2, add = TRUE)
plot(prior_bernoulli(0.4), xlim = c(0, 15))
plot(prior_beta(2, 2))
plot(prior_cauchy(0, 1), xlim = c(-20, 20))
plot(prior_cauchy(0, 2), xlim = c(-20, 20), col = 2, add = TRUE)
plot(prior_gamma(0.1, 0.1))
plot(prior_half_normal(0, 1), xlim = c(-20, 20))
plot(prior_half_normal(0, 2), xlim = c(-20, 20), col = 2, add = TRUE)
plot(prior_poisson(5), xlim = c(0, 15))
plot(uniform_prior(1, 2), xlim = c(0, 3))
```

---

plot\_pdf

*Plot Probability Density Function Values*


---

**Description**

Plot Probability Density Function Values

**Usage**

```
plot_pdf(x, y, ...)
```

**Arguments**

x	values
y	probability density values $y = f(x)$
...	passed to <code>plot()</code>

Plots the density values as a curve with the lower vertical limit set to 0.

**Value**

No return value, this function generates a plot in the current graphics device.

**Examples**

```
x <- seq(-2, 2, len = 100)
y <- dnorm(x)
plot_pdf(x, y)
```

---

plot_pmf	<i>Plot Probability Mass Function Values</i>
----------	--

---

**Description**

Plot Probability Mass Function Values

**Usage**

```
plot_pmf(x, y, ..., col = "grey", add = FALSE)
```

**Arguments**

x	values
y	probability mass values $y = f(x)$
...	passed to <code>plot()</code> and <code>rect()</code>
col	Fill color of bars.
add	Add bars to existing plot. Plots the probability values as a barplot.

**Value**

No return value, this function generates a plot in the current graphics device.

**Examples**

```
x <- seq(0, 5)
y <- dpois(x, lambda = 2)
plot_pmf(x, y)
```

---

poisson_prior	<i>Legacy function for the poisson prior</i>
---------------	--

---

**Description**

Please use `prior_poisson()` instead.

**Usage**

```
poisson_prior(...)
```

**Arguments**

...	Deprecated arguments to <code>poisson_prior()</code> .
-----	--

**Value**

This function does not return a value. When called, it triggers an error message indicating that `poisson_prior()` is deprecated and that `prior_poisson()` should be used instead.

---

possible\_data\_sim\_vars

*Get All Variable Names in Simulated Data Model Matrix*

---

**Description**

Get All Variable Names in Simulated Data Model Matrix

**Usage**

```
possible_data_sim_vars(object)
```

**Arguments**

object            BaselineObject

**Value**

A vector of variable names

---

Prior-class

Prior *Class*

---

**Description**

A class for defining priors to be translated to Stan code. Objects of class `Prior` should not be created directly but by one of the specific prior class constructors.

**Slots**

`stan_code` character. Stan implementation of the prior, with placeholders for parameters surrounded with `{{` and `}}` to be replaced with `glue::glue()`.

`n_param` integer. Number of prior parameters.

`constraint` character. Support of prior distribution expressed as a Stan constraint, e.g. "`<lower=0, upper=1>`".

**See Also**

Prior constructor functions: [prior\\_bernoulli\(\)](#), [prior\\_beta\(\)](#), [prior\\_cauchy\(\)](#), [prior\\_half\\_cauchy\(\)](#), [prior\\_gamma\(\)](#), [prior\\_normal\(\)](#), [prior\\_poisson\(\)](#), [uniform\\_prior\(\)](#)

Other prior classes: [PriorBernoulli-class](#), [PriorBeta-class](#), [PriorCauchy-class](#), [PriorExponential-class](#), [PriorGamma-class](#), [PriorHalfCauchy-class](#), [PriorHalfNormal-class](#), [PriorNormal-class](#), [PriorPoisson-class](#), [UniformPrior-class](#)

---

PriorBernoulli-class    *PriorBernoulli Class*

---

### Description

A class for defining bernoulli priors to be translated to Stan code. Objects of class `PriorBernoulli` should not be created directly but by the constructor `prior_bernoulli()`.

### Slots

`stan_code` character. Stan implementation of the prior, with placeholders for bernoulli stan function parameters surrounded with `{ { and } }` to be replaced with `glue::glue()`.

`n_param` integer. Number of prior parameters (1).

`constraint` character. Support of prior distribution, "`<lower=0, upper=1>`".

`theta` numeric. Probability (in  $[0, 1]$ ).

### See Also

Other prior classes: [Prior-class](#), [PriorBeta-class](#), [PriorCauchy-class](#), [PriorExponential-class](#), [PriorGamma-class](#), [PriorHalfCauchy-class](#), [PriorHalfNormal-class](#), [PriorNormal-class](#), [PriorPoisson-class](#), [UniformPrior-class](#)

---

PriorBeta-class    *PriorBeta Class*

---

### Description

A class for defining beta priors to be translated to Stan code. Objects of class `PriorBeta` should not be created directly but by the constructor `prior_beta()`.

### Slots

`stan_code` character. Stan implementation of the prior, with placeholders for beta stan function parameters surrounded with `{ { and } }` to be replaced with `glue::glue()`.

`n_param` integer. Number of prior parameters (2).

`constraint` character. Support of prior distribution, "`<lower=0, upper=1>`".

`alpha` numeric. Shape ( $\geq 0$ ).

`beta` numeric. Shape ( $\geq 0$ ).

### See Also

Other prior classes: [Prior-class](#), [PriorBernoulli-class](#), [PriorCauchy-class](#), [PriorExponential-class](#), [PriorGamma-class](#), [PriorHalfCauchy-class](#), [PriorHalfNormal-class](#), [PriorNormal-class](#), [PriorPoisson-class](#), [UniformPrior-class](#)

---

PriorCauchy-class	PriorCauchy Class
-------------------	-------------------

---

**Description**

A class for defining the cauchy priors to be translated to Stan code. Objects of class `PriorCauchy` should not be created directly but by the constructor `prior_cauchy()`.

**Slots**

`stan_code` character. Stan implementation of the prior, with placeholders for cauchy stan function parameters surrounded with `{{` and `}}` to be replaced with `glue::glue()`.

`n_param` integer. Number of prior parameters (2).

`constraint` character. Support of prior distribution, (all values allowed in cauchy distribution).

`mu` numeric. Location.

`sigma` numeric. Scale (>0).

**See Also**

Other prior classes: [Prior-class](#), [PriorBernoulli-class](#), [PriorBeta-class](#), [PriorExponential-class](#), [PriorGamma-class](#), [PriorHalfCauchy-class](#), [PriorHalfNormal-class](#), [PriorNormal-class](#), [PriorPoisson-class](#), [UniformPrior-class](#)

---

PriorExponential-class	PriorExponential Class
------------------------	------------------------

---

**Description**

A class for defining exponential priors to be translated to Stan code. Objects of class `PriorExponential` should not be created directly but by the constructor `prior_exponential()`.

**Slots**

`stan_code` character. Stan implementation of the prior, with placeholders for exponential Stan function parameters surrounded with `{{` and `}}` to be replaced with `glue::glue()`.

`n_param` integer. Number of prior parameters (1).

`constraint` character. Support of prior distribution, "`<lower=0>`".

`beta` numeric. Inverse scale (>0).

**See Also**

Other prior classes: [Prior-class](#), [PriorBernoulli-class](#), [PriorBeta-class](#), [PriorCauchy-class](#), [PriorGamma-class](#), [PriorHalfCauchy-class](#), [PriorHalfNormal-class](#), [PriorNormal-class](#), [PriorPoisson-class](#), [UniformPrior-class](#)

---

PriorGamma-class      *PriorGamma Class*

---

### Description

A class for defining gamma priors to be translated to Stan code. Objects of class `PriorGamma` should not be created directly but by the constructor `prior_gamma()`.

### Slots

`stan_code` character. Stan implementation of the prior, with placeholders for gamma stan function parameters surrounded with `{{` and `}}` to be replaced with `glue::glue()`.

`n_param` integer. Number of prior parameters (2).

`constraint` character. Support of prior distribution, "`<lower=0>`".

`alpha` numeric. Shape ( $>0$ ).

`beta` numeric. Inverse scale ( $\geq 0$ ).

### See Also

Other prior classes: [Prior-class](#), [PriorBernoulli-class](#), [PriorBeta-class](#), [PriorCauchy-class](#), [PriorExponential-class](#), [PriorHalfCauchy-class](#), [PriorHalfNormal-class](#), [PriorNormal-class](#), [PriorPoisson-class](#), [UniformPrior-class](#)

---

PriorHalfCauchy-class      *PriorHalfCauchy Class*

---

### Description

A class for defining half cauchy priors to be translated to Stan code. Objects of class `PriorHalfCauchy` should not be created directly but by the constructor `prior_half_cauchy()`.

### Slots

`stan_code` character. Stan implementation of the prior, with placeholders for the half cauchy stan function parameters surrounded with `{{` and `}}` to be replaced with `glue::glue()`.

`n_param` integer. Number of prior parameters (2).

`constraint` character. Support of prior distribution. In a half cauchy prior, constraint is `mu`

`mu` numeric. Location.

`sigma` numeric. Scale ( $>0$ ).

### See Also

Other prior classes: [Prior-class](#), [PriorBernoulli-class](#), [PriorBeta-class](#), [PriorCauchy-class](#), [PriorExponential-class](#), [PriorGamma-class](#), [PriorHalfNormal-class](#), [PriorNormal-class](#), [PriorPoisson-class](#), [UniformPrior-class](#)

---

PriorHalfNormal-class    *PriorHalfNormal Class*

---

### Description

A class for defining half normal priors to be translated to Stan code. Objects of class `PriorHalfNormal` should not be created directly but by the constructor `prior_half_normal()`.

### Slots

`stan_code` character. Stan implementation of the prior, with placeholders for the half normal stan function parameters surrounded with `{{` and `}}` to be replaced with `glue::glue()`.

`n_param` integer. Number of prior parameters (2).

`constraint` character. Support of prior distribution. In a half normal prior, constraint is `mu`

`mu` numeric. Location.

`sigma` numeric. Scale (>0).

### See Also

Other prior classes: [Prior-class](#), [PriorBernoulli-class](#), [PriorBeta-class](#), [PriorCauchy-class](#), [PriorExponential-class](#), [PriorGamma-class](#), [PriorHalfCauchy-class](#), [PriorNormal-class](#), [PriorPoisson-class](#), [UniformPrior-class](#)

---

PriorNormal-class    *PriorNormal Class*

---

### Description

A class for defining normal priors to be translated to Stan code. Objects of class `PriorNormal` should not be created directly but by the constructor `prior_normal()`.

### Slots

`stan_code` character. Stan implementation of the prior, with placeholders for normal stan function parameters surrounded with `{{` and `}}` to be replaced with `glue::glue()`.

`n_param` integer. Number of prior parameters (2).

`constraint` character. Support of prior distribution, (all values allowed in normal distribution).

`mu` numeric. Location.

`sigma` numeric. Scale (>0).

### See Also

Other prior classes: [Prior-class](#), [PriorBernoulli-class](#), [PriorBeta-class](#), [PriorCauchy-class](#), [PriorExponential-class](#), [PriorGamma-class](#), [PriorHalfCauchy-class](#), [PriorHalfNormal-class](#), [PriorPoisson-class](#), [UniformPrior-class](#)

---

PriorPoisson-class      *PriorPoisson Class*

---

**Description**

A class for defining poisson priors to be translated to Stan code. Objects of class `PriorPoisson` should not be created directly but by the constructor `prior_poisson()`.

**Slots**

`stan_code` character. Stan implementation of the prior, with placeholders for poisson stan function parameters surrounded with `{{` and `}}` to be replaced with `glue::glue()`.

`n_param` integer. Number of prior parameters (1).

`constraint` character. Support of prior distribution, "`<lower=0>`".

`lambda` numeric. Rate ( $>0$ ).

**See Also**

Other prior classes: [Prior-class](#), [PriorBernoulli-class](#), [PriorBeta-class](#), [PriorCauchy-class](#), [PriorExponential-class](#), [PriorGamma-class](#), [PriorHalfCauchy-class](#), [PriorHalfNormal-class](#), [PriorNormal-class](#), [UniformPrior-class](#)

---

prior\_bernoulli      *Prior bernoulli distribution*

---

**Description**

Prior bernoulli distribution

**Usage**

```
prior_bernoulli(theta)
```

**Arguments**

`theta`                  numeric. Probability (in  $[0, 1]$ ).

**Details**

Stan reference <https://mc-stan.org/docs/functions-reference/bernoulli-distribution.html>

**Value**

Object of class `PriorBernoulli`.

**See Also**

Other priors: [prior\\_beta\(\)](#), [prior\\_cauchy\(\)](#), [prior\\_exponential\(\)](#), [prior\\_gamma\(\)](#), [prior\\_half\\_cauchy\(\)](#), [prior\\_half\\_normal\(\)](#), [prior\\_normal\(\)](#), [prior\\_poisson\(\)](#), [uniform\\_prior\(\)](#)

**Examples**

```
bp <- prior_bernoulli(0.23)
```

---

prior_beta	<i>Prior beta distribution</i>
------------	--------------------------------

---

**Description**

Prior beta distribution

**Usage**

```
prior_beta(alpha, beta)
```

**Arguments**

alpha	numeric. Shape ( $\geq 0$ ).
beta	numeric. Shape ( $\geq 0$ ).

**Details**

Stan reference <https://mc-stan.org/docs/functions-reference/beta-distribution.html>

**Value**

Object of class [PriorBeta](#)

**See Also**

Other priors: [prior\\_bernoulli\(\)](#), [prior\\_cauchy\(\)](#), [prior\\_exponential\(\)](#), [prior\\_gamma\(\)](#), [prior\\_half\\_cauchy\(\)](#), [prior\\_half\\_normal\(\)](#), [prior\\_normal\(\)](#), [prior\\_poisson\(\)](#), [uniform\\_prior\(\)](#)

**Examples**

```
bp <- prior_beta(9, 235)
```

---

prior\_cauchy      *Prior cauchy distribution*

---

**Description**

Prior cauchy distribution

**Usage**

```
prior_cauchy(mu, sigma)
```

**Arguments**

mu                  numeric. Location.  
sigma                numeric. Scale (>0).

**Details**

Stan reference <https://mc-stan.org/docs/functions-reference/cauchy-distribution.html>

**Value**

Object of class `PriorCauchy`.

**See Also**

Other priors: `prior_bernoulli()`, `prior_beta()`, `prior_exponential()`, `prior_gamma()`, `prior_half_cauchy()`, `prior_half_normal()`, `prior_normal()`, `prior_poisson()`, `uniform_prior()`

**Examples**

```
cp <- prior_cauchy(1, 1)
```

---

prior\_exponential      *Prior exponential distribution*

---

**Description**

Prior exponential distribution

**Usage**

```
prior_exponential(beta)
```

**Arguments**

beta                numeric. Inverse scale (>0).

**Details**

Stan reference <https://mc-stan.org/docs/functions-reference/exponential-distribution.html>

**Value**

Object of class `PriorExponential`.

**See Also**

Other priors: `prior_bernoulli()`, `prior_beta()`, `prior_cauchy()`, `prior_gamma()`, `prior_half_cauchy()`, `prior_half_normal()`, `prior_normal()`, `prior_poisson()`, `uniform_prior()`

**Examples**

```
ep <- prior_exponential(1)
```

---

prior_gamma	<i>Prior gamma distribution</i>
-------------	---------------------------------

---

**Description**

Prior gamma distribution

**Usage**

```
prior_gamma(alpha, beta)
```

**Arguments**

alpha	numeric. Shape ( $>0$ ).
beta	numeric. Inverse scale ( $\geq 0$ ).

**Details**

Stan reference <https://mc-stan.org/docs/functions-reference/gamma-distribution.html>

**Value**

Object of class `PriorGamma`.

**See Also**

Other priors: `prior_bernoulli()`, `prior_beta()`, `prior_cauchy()`, `prior_exponential()`, `prior_half_cauchy()`, `prior_half_normal()`, `prior_normal()`, `prior_poisson()`, `uniform_prior()`

**Examples**

```
gp <- prior_gamma(0.001, 0.001)
```

---

prior\_half\_cauchy      *Prior half-cauchy distribution*

---

**Description**

Prior half-cauchy distribution

**Usage**

```
prior_half_cauchy(mu, sigma)
```

**Arguments**

mu                    numeric. Location.  
sigma                 numeric. Scale (>0).

**Details**

Stan reference <https://mc-stan.org/docs/functions-reference/cauchy-distribution.html>

**Value**

Object of class `PriorHalfCauchy`.

**See Also**

Other priors: `prior_bernoulli()`, `prior_beta()`, `prior_cauchy()`, `prior_exponential()`, `prior_gamma()`, `prior_half_normal()`, `prior_normal()`, `prior_poisson()`, `uniform_prior()`

**Examples**

```
hpc <- prior_half_cauchy(1, 1)
```

---

prior\_half\_normal      *Prior half-normal distribution*

---

**Description**

Prior half-normal distribution

**Usage**

```
prior_half_normal(mu, sigma)
```

**Arguments**

mu                numeric. Location.  
sigma             numeric. Scale (>0).

**Details**

Stan reference <https://mc-stan.org/docs/functions-reference/normal-distribution.html>

**Value**

Object of class `PriorHalfNormal`.

**See Also**

Other priors: [prior\\_bernoulli\(\)](#), [prior\\_beta\(\)](#), [prior\\_cauchy\(\)](#), [prior\\_exponential\(\)](#), [prior\\_gamma\(\)](#), [prior\\_half\\_cauchy\(\)](#), [prior\\_normal\(\)](#), [prior\\_poisson\(\)](#), [uniform\\_prior\(\)](#)

**Examples**

```
hcp <- prior_half_normal(1, 1)
```

---

prior_normal	<i>Prior normal distribution</i>
--------------	----------------------------------

---

**Description**

Prior normal distribution

**Usage**

```
prior_normal(mu, sigma)
```

**Arguments**

mu                numeric. Location.  
sigma             numeric. Scale (>0).

**Details**

Stan reference <https://mc-stan.org/docs/functions-reference/normal-distribution.html>

**Value**

Object of class `PriorNormal`.

**See Also**

Other priors: [prior\\_bernoulli\(\)](#), [prior\\_beta\(\)](#), [prior\\_cauchy\(\)](#), [prior\\_exponential\(\)](#), [prior\\_gamma\(\)](#), [prior\\_half\\_cauchy\(\)](#), [prior\\_half\\_normal\(\)](#), [prior\\_poisson\(\)](#), [uniform\\_prior\(\)](#)

**Examples**

```
np <- prior_normal(1, 1)
```

---

prior_poisson	<i>Prior poisson distribution</i>
---------------	-----------------------------------

---

**Description**

Prior poisson distribution

**Usage**

```
prior_poisson(lambda)
```

**Arguments**

lambda            numeric. Rate (>0).

**Details**

Stan reference <https://mc-stan.org/docs/functions-reference/poisson.html>

**Value**

Object of class [PriorPoisson](#).

**See Also**

Other priors: [prior\\_bernoulli\(\)](#), [prior\\_beta\(\)](#), [prior\\_cauchy\(\)](#), [prior\\_exponential\(\)](#), [prior\\_gamma\(\)](#), [prior\\_half\\_cauchy\(\)](#), [prior\\_half\\_normal\(\)](#), [prior\\_normal\(\)](#), [uniform\\_prior\(\)](#)

**Examples**

```
pp <- prior_poisson(100)
```

---

`rename_draws_covariates`*Rename Covariates in draws Object*

---

**Description**

Rename Covariates in draws Object

**Usage**

```
rename_draws_covariates(draws, analysis)
```

**Arguments**

`draws` draws created from sampled analysis object. See example.  
`analysis` Analysis as created by `create_analysis_obj()`.

**Value**

A `draws[posterior::draws]` object with covariate names.

**Examples**

```
if (check_cmdstan()) {  
  analysis_object <- create_analysis_obj(  
    data_matrix = example_matrix,  
    covariates = add_covariates(  
      covariates = c("cov1", "cov2"),  
      priors = prior_normal(0, 1000)  
    ),  
    outcome = outcome_surv_exponential(  
      "time",  
      "cnsr",  
      baseline_prior = prior_normal(0, 1000)  
    ),  
    borrowing = borrowing_hierarchical_commensurate(  
      "ext",  
      prior_exponential(.001)  
    ),  
    treatment = treatment_details(  
      "trt",  
      prior_normal(0, 1000)  
    )  
  )  
  samples <- mcmc_sample(analysis_object, 200, 400, 1)  
  draws <- samples$draws()  
  renamed_draws <- rename_draws_covariates(draws, analysis_object)  
  summary(renamed_draws)  
}
```

---

set_cut_off	<i>Set Clinical Cut Off Rule</i>
-------------	----------------------------------

---

**Description**

Set Clinical Cut Off Rule

**Usage**

```
set_cut_off(object, internal = cut_off_none(), external = cut_off_none())
```

**Arguments**

object	DataSimObject
internal	DataSimCutOff object specified by one of the cut off functions: cut_off_after_events(), cut_off_after_first(), cut_off_after_last(), cut_off_none().
external	DataSimCutOff for the external data.

**Value**

A DataSimObject with updated cut\_off\_internal and cut\_off\_external slots.

**Examples**

```
data_sim <- create_data_simulation(
  create_baseline_object(10, 10, 10),
  event_dist = create_event_dist(dist = "exponential", lambdas = 1 / 36)
)
set_cut_off(
  data_sim,
  cut_off_after_events(n = 10),
  cut_off_after_first(time = 30)
)
```

---

set_dropout	<i>Set Drop Out Distribution</i>
-------------	----------------------------------

---

**Description**

Set Drop Out Distribution

**Usage**

```
set_dropout(object, internal_treated, internal_control, external_control)
```

**Arguments**

object	DataSimObject
internal_treated	DataSimEvent object specifying distribution for internal treated patients.
internal_control	DataSimEvent object specifying distribution for internal control patients.
external_control	DataSimEvent object specifying distribution for external control patients.

**Details**

DataSimEvent objects can be specified with `create_event_dist()`. Currently no beta parameters can be used in drop out distributions (unlike for the survival outcome).

**Value**

A DataSimObject with updated `internal_treated`, `internal_control` and `external_control` slots.

**Examples**

```
data_sim <- create_data_simulation(
  create_baseline_object(10, 10, 10),
  event_dist = create_event_dist(dist = "exponential", lambdas = 1 / 36)
)
set_dropout(
  data_sim,
  internal_treated = create_event_dist(dist = "exponential", lambdas = 1 / 55),
  internal_control = create_event_dist(dist = "exponential", lambdas = 1 / 50),
  external_control = create_event_dist(dist = "exponential", lambdas = 1 / 40)
)
```

---

set_enrollment	<i>Set Enrollment Rates for Internal and External Trials</i>
----------------	--

---

**Description**

Set Enrollment Rates for Internal and External Trials

**Usage**

```
set_enrollment(object, internal, external = internal)
```

**Arguments**

object	A DataSimObject from <code>create_data_simulation</code>
internal	DataSimEnrollment object to define the enrollment times for internal data
external	DataSimEnrollment object to define the enrollment times for external data. Defaults to be the same as internal.

**Value**

A DataSimObject with updated enrollment\_internal and enrollment\_external slots.

**Examples**

```
data_sim <- create_data_simulation(  
  create_baseline_object(10, 10, 10),  
  event_dist = create_event_dist(dist = "exponential", lambdas = 1 / 36)  
)  
set_enrollment(  
  data_sim,  
  internal = enrollment_constant(rate = c(10, 5), for_time = c(6, 6)),  
  external = enrollment_constant(rate = c(5), for_time = c(20))  
)
```

---

set\_transformations    *Set transformations in BaselineObject objects*

---

**Description**

Set transformations in BaselineObject objects

**Usage**

```
set_transformations(object, ..., overwrite = FALSE)
```

**Arguments**

object	BaselineObject object
...	Additional arguments passed to methods
overwrite	logical. Overwrite existing transformations?

**Value**

BaselineObject object with transformations

---

set\_transformations,BaselineObject-method  
*Set Transformations in Baseline Objects*

---

## Description

Set Transformations in Baseline Objects

## Usage

```
## S4 method for signature 'BaselineObject'  
set_transformations(object, ..., overwrite = FALSE)
```

## Arguments

object	BaselineObject created by <a href="#">create_baseline_object</a> .
...	named transformation functions. See details for more information.
overwrite	If TRUE overwrite existing transformation list and only include newly specified transformations.

## Details

Transformation functions are evaluated in order and create or overwrite a column in the data.frame with that name. The function should have signature `function(data)`, taking a data.frame (specifically a `BaselineDataFrame` object from `generate(BaselineObject)`) and return a vector with length identical to the total number of patients. The `@BaselineObject` slot of the `BaselineDataFrame` may be accessed directly or with `get_quantiles()` to create transformations. See `binary_cutoff()`.

## Value

An updated `BaselineObject`

## Examples

```
baseline <- create_baseline_object(  
  100, 50, 100,  
  covariates = baseline_covariates(  
    names = "age", means_int = 55,  
    covariance_int = covariance_matrix(5)  
  )  
)  
set_transformations(baseline, age_scaled = function(data) scale(data$age))
```

---

show_guide	<i>Show guide for objects with guides</i>
------------	---

---

**Description**

Show the guide in Simulation objects.

**Usage**

```
show_guide(object)
```

```
## S4 method for signature 'Simulation'
show_guide(object)
```

**Arguments**

object            Simulation object

**Value**

A data.frame showing all simulation scenarios.

---

SimBorrowingList-class	<i>SimBorrowingList Class</i>
------------------------	-------------------------------

---

**Description**

A class for borrowing details as part of a simulation study. Objects of class `SimBorrowingList` should not be created directly but by the constructor `sim_borrowing_list()`.

**Slots**

`borrowing_list` named list of object of class `Borrowing`, one object for each parameter variation.

---

SimCovariateList-class	<i>SimCovariateList Class</i>
------------------------	-------------------------------

---

**Description**

A class for covariate details as part of a simulation study. Objects of class `SimCovariateList` should not be created directly but by the constructor `sim_covariate_list()`.

**Slots**

`covariate_list` named list of object of class `Covariate`, one object for each parameter variation.

---

SimCovariates-class      *SimCovariates Class*

---

### Description

A class for specifying covariate distributions and covariance for simulation studies.

### Slots

`covariates` list. List of covariate mean values or probabilities as generated through `bin_var()` (class `SimVarBin` or `cont_var()` (class `SimVarCont`).

`covariance_internal` matrix. Covariance matrix before binarization for internal patients.

`covariance_external` matrix. Covariance matrix before binarization for external patients.

---

SimDataList-class      *SimDataList Class*

---

### Description

A class for defining generated data for use as part of a simulation study. Objects of class `SimDataList` should not be created directly but by the constructor `sim_data_list()`.

### Slots

`data_list` list of lists of matrices. The lists at the highest level differ in that the parameters used to generate the data. The matrices at lowest level are different iterations of the same data generation parameters.

`guide` data.frame. `guide` contains information on the parameters that differ at the highest level of `data_list`.

`effect` character. The column in `guide` that corresponds to the true treatment effect estimate (hazard ratio or odds ratio).

`drift` character. The column in `guide` that corresponds to the drift between external and internal control arms. A drift >1 means the external arm experiences greater effects.

`index` character. The column in `guide` that corresponds to the index of the parameter situations in `data_list`.

---

SimOutcomeList-class    *SimOutcomeList Class*

---

### Description

A class for outcome details as part of a simulation study. Objects of class `SimOutcomeList` should not be created directly but by the constructor `sim_outcome_list()`.

### Slots

`outcome_list` named list of object of class `Outcome`, one object for each parameter variation.

---

SimSampleSize-class    *SimSampleSize Class*

---

### Description

A class for creating matrices for simulation studies containing flags specifying whether the patient is from the concurrent trial or not (`ext = 0` for concurrent trial, `ext = 1` for historical data) and whether the patient is on the experimental therapy or not (`trt = 0` for no experimental therapy, `trt = 1` for experimental therapy).

### Slots

`n_internal_control` integer. Number of patients to be simulated in the internal control arm.  
`n_external_control` integer. Number of patients to be simulated in the external control arm.  
`n_internal_experimental` integer. Number of patients to be simulated in the internal experimental arm.  
`mat` matrix. Matrix with two columns, `ext` (flag for being from external data source) and `trt` (flag for receiving experimental treatment)

---

SimTreatmentList-class  
    *SimTreatmentList Class*

---

### Description

A class for treatment details as part of a simulation study. Objects of class `SimTreatmentList` should not be created directly but by the constructor `sim_treatment_list()`.

### Slots

`treatment_list` named list of object of class `Treatment`, one object for each parameter variation.

---

Simulation-class	Simulation Class
------------------	------------------

---

**Description**

A class for defining Simulation study details. Objects of class `Simulation` should not be created directly but by the constructor `create_simulation_obj()`.

**Slots**

`data_matrix_list` `SimDataList`. The list of lists of data matrices created with `sim_data_list()`.  
`outcome` `SimOutcomeList`. List of Outcome objects created with `sim_outcome_list()`.  
`borrowing` `SimBorrowingList`. List of Borrowing objects created with `sim_borrowing_list()`.  
`covariate` `SimCovariateList` or `NULL`. List of Covariate objects created with `sim_covariate_list()` or `NULL` (no covariate adjustment).  
`treatment` `SimTreatmentList`. List of Treatment objects created with `sim_treatment_list()`.  
`guide` `data.frame`. Data.frame containing information on all combinations evaluated.  
`n_combos` `integer`. Number of combinations of parameters to be evaluated.  
`n_analyses` `integer`. Number of analyses (combos x datasets to be performed).  
`'analysis_obj_list'` `list`. List of analysis objects indexed according to `guide`.

---

SimVar-class	SimVar Class
--------------	--------------

---

**Description**

A parent class for defining covariates to be created in the simulation study calls to `add_covariates()`.

---

SimVarBin-class	SimVarBin class
-----------------	-----------------

---

**Description**

A constructor for making objects of class `SimVarBin`. Objects of class `SimVarBin` are used to hold proportions of binary variables specified in a simulation study.

**Slots**

prob\_internal numeric. Proportion for the internal arms.  
prob\_external numeric. Proportion for the external arm.  
mu\_internal\_before\_bin numeric. Mean value of covariate before binarization for the internal arms.  
mu\_external\_before\_bin numeric. Mean value of covariate before binarization for the external arm.  
printval\_int numeric. Value to print to summarize internal arms.  
printval\_ext numeric. Value to print to summarize external arm.  
type\_string character. 'binary'

**See Also**

Other simvar classes: [SimVarCont-class](#)

---

SimVarCont-class	SimVarCont class
------------------	------------------

---

**Description**

A constructor for making objects of class SimVarCont. Objects of class SimVarCont are used to hold mean values of of continuous variables specified in a simulation study.

**Slots**

mu\_internal numeric. Mean covariate value for the internal arms.  
mu\_external numeric. Mean covariate value for the external arm.  
printval\_int numeric. Value to print to summarize internal arms.  
printval\_ext numeric. Value to print to summarize external arm.  
type\_string character. 'continuous'

**See Also**

Other simvar classes: [SimVarBin-class](#)

---

sim\_borrowing\_list     *Input borrowing details for a simulation study*

---

## Description

A function for defining which borrowing scenarios should be evaluated as part of a simulation study.

## Usage

```
sim_borrowing_list(borrowing_list)
```

## Arguments

`borrowing_list` named list of objects of class `Borrowing` created by `borrowing_full()`, `borrowing_none()`, or `borrowing_hierarchical_commensurate()`.

## Value

Object of class `SimBorrowingList`.

## See Also

Other simulation classes: `sim_covariate_list()`, `sim_data_list()`, `sim_outcome_list()`, `sim_treatment_list()`

## Examples

```
borrow_scenarios <- sim_borrowing_list(  
  list(  
    "No borrowing" = borrowing_none("ext"),  
    "Full borrowing" = borrowing_full("ext"),  
    "BDB, uninformative prior" = borrowing_hierarchical_commensurate(  
      "ext",  
      prior_gamma(0.001, 0.001)  
    ),  
    "BDB, informative prior" = borrowing_hierarchical_commensurate(  
      "ext",  
      prior_gamma(1, 0.001)  
    )  
  )  
)  
)
```

---

sim\_covariates      *Specify covariates for simulation study*

---

## Description

Provide details on the desired covariate distributions and covariance for for a simulation study.

## Usage

```
sim_covariates(  
  covariates,  
  covariance_internal,  
  covariance_external = covariance_internal  
)
```

## Arguments

**covariates**      list. Named list of covariate mean values or probabilities as generated through `bin_var()` (class `SimVarBin` or `cont_var()` (class `SimVarCont`). See details for more information.

**covariance\_internal**  
matrix. Covariance matrix before binarization for internal patients.

**covariance\_external**  
matrix. Covariance matrix before binarization for external patients. Defaults to the internal covariance.

## Details

This function is intended to specify the number of covariates and relationships between them for the purposes of designing a simulation study in `psborrow2`. Because the outcome model does not necessarily need to adjust for covariates, this function is not necessary in `create_simulation_obj()`. The relationship between the treatment and the outcome is specified elsewhere (i.e, in `sim_survival()` or `sim_binary_event()`).

We need a few things to

## Value

Object of class `SimCovariates`

## See Also

Other simulation: [sim\\_samplesize\(\)](#)

**Examples**

```

set.seed(123)
covmat <- matrix(rWishart(1, 2, diag(2)), ncol = 2)

covset1 <- sim_covariates(
  covariates = list(
    cov1 = bin_var(0.5, 0.5),
    cov2 = cont_var(100, 130)
  ),
  covariance_internal = covmat
)

```

---

sim_covariates_summ	<i>Summarize the number of continuous and binary covariates in a SimCovariates object created by sim_covariates()</i>
---------------------	---

---

**Description**

Summarize the number of continuous and binary covariates in a SimCovariates object created by sim\_covariates()

**Usage**

```
sim_covariates_summ(sim_covariates_obj)
```

**Arguments**

sim\_covariates\_obj  
 SimCovariates. Object returned by sim\_covariates().

**Value**

data.frame showing covariate names and types as well as counts of binary and continuous covariates.

---

sim_covariate_list	<i>Input covariate adjustment details for a simulation study</i>
--------------------	--

---

**Description**

A function for defining which covariate adjustment scenarios should be evaluated as part of a simulation study.

**Usage**

```
sim_covariate_list(covariate_list)
```

**Arguments**

covariate\_list named list of objects of class Covariate created by add\_covariates().

**Details**

This function allows the user to specify covariate adjustment details that will be included as part of a simulation study. It is often of interest to compare several adjustment methods to no adjustment. To specify no adjustment, pass NULL as a list item to covariate\_list.

**Value**

Object of class `SimCovariateList`.

**See Also**

Other simulation classes: `sim_borrowing_list()`, `sim_data_list()`, `sim_outcome_list()`, `sim_treatment_list()`

**Examples**

```
covariates <- sim_covariate_list(
  list(
    "No adjustment" = NULL,
    "Covariates 1 and 2" = add_covariates(c("cov1", "cov2"), prior_normal(0, 1000))
  )
)
```

---

sim\_data\_list

*Input generated data for a simulation study*

---

**Description**

A function for defining generated data for use as part of a simulation study.

**Usage**

```
sim_data_list(data_list, guide, effect, drift, index)
```

**Arguments**

data_list	list of lists of matrices. The lists at the highest level differ in that the parameters used to generate the data. The matrices at lowest level are different iterations of the same data generation parameters. See details.
guide	data.frame. guide contains information on the parameters that differ at the highest level of data_list. See details.
effect	character. The column in guide that corresponds to the true treatment effect estimate (hazard ratio or odds ratio).

drift	character. The column in guide that corresponds to the true drift effect estimate (hazard ratio or odds ratio). A drift >1 means the external arm experiences greater effects.
index	character. The column in guide that corresponds to the index column.

## Details

In this function, you are providing generated data for analysis in a simulation study in `psborrow2`. Note that this function does not do any data generation on your behalf; it assumes that you have generated the data already. For a full working example, refer to the relevant vignette: `vignette('simulation_study', package = 'psborrow2')`.

More information on the inputs is provided below.

### Matrix requirements in `data_list`:

Each matrix embedded in `data_list` must have:

1. a flag for whether the patient is an external control
2. a flag for whether the patient is in the experimental treatment arm
3. outcome information (time and censorship for survival, flag for outcome in binary endpoints)

Optionally, the matrices may also contain covariates. See examples.

### `data_list`:

Each set of distinct data generation parameters should be represented by a single list of matrices. Because multiple scenarios may want to be compared, a list of list of matrices is preferred. See examples.

### `guide`:

The guide should be a `data.frame` with one row per scenario. As a consequence of this, the length of the list should equal the number of rows in the guide. See examples.

## Value

Object of class `SimDataList`.

## See Also

Other simulation classes: `sim_borrowing_list()`, `sim_covariate_list()`, `sim_outcome_list()`, `sim_treatment_list()`

## Examples

```
base_mat <- matrix(
  c(
    rep(0, 200), rep(0, 200), rep(1, 200),
    rep(1, 200), rep(0, 200), rep(0, 200),
    rep(0, 600)
  ),
  ncol = 3,
  dimnames = list(NULL, c("ext", "trt", "driftOR"))
)
```

```

add_binary_endpoint <- function(odds_ratio,
                                base_matrix = base_mat) {
  linear_predictor <- base_matrix[, "trt"] * log(odds_ratio)
  prob <- 1 / (1 + exp(-linear_predictor))

  bin_endpoint <- rbinom(
    NROW(base_matrix),
    1,
    prob
  )

  cbind(base_matrix, matrix(bin_endpoint, ncol = 1, dimnames = list(NULL, "ep")))
}

data_list <- list(
  list(add_binary_endpoint(1.5), add_binary_endpoint(1.5)),
  list(add_binary_endpoint(2.5), add_binary_endpoint(2.5))
)

guide <- data.frame(
  trueOR = c(1.5, 2.5),
  driftOR = c(1.0, 1.0),
  ind = c(1, 2)
)

sdl <- sim_data_list(
  data_list = data_list,
  guide = guide,
  effect = "trueOR",
  drift = "driftOR",
  index = "ind"
)

```

---

sim\_outcome\_list      *Input outcome details for a simulation study*

---

### Description

A function for defining which outcome scenarios should be evaluated as part of a simulation study.

### Usage

```
sim_outcome_list(outcome_list)
```

### Arguments

outcome\_list      named list of objects of class Outcome created by outcome\_details().

**Value**

Object of class `SimOutcomeList`.

**See Also**

Other simulation classes: `sim_borrowing_list()`, `sim_covariate_list()`, `sim_data_list()`, `sim_treatment_list()`

**Examples**

```
outcome_scenarios <- sim_outcome_list(
  list(
    "Exponential" = outcome_surv_exponential("time", "cnsr", prior_normal(0, 10000))
  )
)
```

---

<code>sim_samplesize</code>	<i>Set simulation study parameters for sample size</i>
-----------------------------	--

---

**Description**

Set simulation study parameters for sample size

**Usage**

```
sim_samplesize(n_internal_control, n_external_control, n_internal_experimental)
```

**Arguments**

`n_internal_control`  
integer. Number of patients to be simulated in the internal control arm.

`n_external_control`  
integer. Number of patients to be simulated in the external control arm.

`n_internal_experimental`  
integer. Number of patients to be simulated in the internal experimental arm.

**Value**

Object of class `SimSampleSize`

**See Also**

Other simulation: `sim_covariates()`

**Examples**

```
ss <- sim_samplesize(200, 200, 500)
```

---

`sim_treatment_list`     *Input treatment details for a simulation study*

---

### Description

A function for defining which treatment scenarios should be evaluated as part of a simulation study.

### Usage

```
sim_treatment_list(treatment_list)
```

### Arguments

`treatment_list` named list of objects of class `Treatment` created by `treatment_details()`.

### Value

Object of class `SimTreatmentList`.

### See Also

Other simulation classes: `sim_borrowing_list()`, `sim_covariate_list()`, `sim_data_list()`, `sim_outcome_list()`

### Examples

```
treatment_scenarios <- sim_treatment_list(  
  list(  
    "Standard" = treatment_details("trt", prior_normal(0, 1000))  
  )  
)
```

---

TimeToEvent-class     TimeToEvent *class*

---

### Description

TimeToEvent class

**Slots**

`n_param` integer. Number of ancillary parameters for the model to estimate.

`param_priors` list. Named list of prior distributions on the ancillary parameters in the model.

`time_var` character. Variable used for time in `TimeToEvent` objects.

`cens_var` character. Variable used for censoring in `TimeToEvent` objects.

`baseline_prior` Prior. Object of class `Prior` specifying prior distribution for the baseline outcome.

`name_beta_trt`. Named vector for `beta_trt`.

`name_exp_trt`. Named vector for exponentiated `beta_trt`

`alpha_type`. How to interpret alpha.

`name_addnl_params`. Named vector for additional parameters.

**See Also**

Other outcome: [BinaryOutcome-class](#), [ContinuousOutcome-class](#), [Outcome-class](#), [OutcomeBinaryLogistic-class](#), [OutcomeContinuousNormal-class](#), [OutcomeSurvExponential-class](#), [OutcomeSurvPEM-class](#), [OutcomeSurvWeibullPH-class](#)

---

Treatment-class

Treatment *Class*


---

**Description**

A class for defining treatment details. Objects of class `Treatment` should not be created directly but by the constructor `treatment_details()`.

**Slots**

`trt_flag_col` character. Character specifying the name of the column in the model matrix that corresponds to the treatment flag (1/0 or TRUE/FALSE). This identifies patients as belonging to the experimental treatment arm.

`trt_prior` Prior. Object of class `Prior` specifying the prior distribution of the log effect estimate (log hazard ratio for time to event endpoints and log odds ratio for binary endpoints).

---

treatment_details	<i>Specify Treatment Details</i>
-------------------	----------------------------------

---

### Description

Specify the treatment arm column name in the model matrix and set a prior distribution for the treatment effect (log hazard ratio or log odds ratio)

### Usage

```
treatment_details(trt_flag_col, trt_prior)
```

### Arguments

trt_flag_col	character. The name of the column in the model matrix that corresponds to the treatment flag (1/0 or TRUE/FALSE). This identifies patients as belonging to the experimental treatment arm.
trt_prior	Object of class <code>Prior</code> specifying the prior distribution of the log effect estimate (log hazard ratio for time to event endpoints and log odds ratio for binary endpoints).

### Value

Object of class `Treatment`.

### Examples

```
sta <- treatment_details(
  trt_flag_col = "trt",
  trt_prior = prior_normal(0, 1000)
)
```

---

trim_cols	<i>Trim columns from Data Matrix Based on Borrowing object type</i>
-----------	---

---

### Description

Trim columns from Data Matrix Based on Borrowing object type

**Usage**

```
trim_cols(borrowing_object, analysis_object)

## S4 method for signature 'Borrowing'
trim_cols(borrowing_object, analysis_object)

## S4 method for signature 'BorrowingFixedPowerPrior'
trim_cols(borrowing_object, analysis_object)

## S4 method for signature 'BorrowingHierarchicalCommensurate'
trim_cols(borrowing_object, analysis_object)
```

**Arguments**

```
borrowing_object
      borrowing object
analysis_object
      analysis object
```

---

trim\_rows

*Trim Rows from Data Matrix Based on Borrowing object type*

---

**Description**

Trim Rows from Data Matrix Based on Borrowing object type

**Usage**

```
trim_rows(borrowing_object, analysis_object)

## S4 method for signature 'Borrowing'
trim_rows(borrowing_object, analysis_object)

## S4 method for signature 'BorrowingNone'
trim_rows(borrowing_object, analysis_object)
```

**Arguments**

```
borrowing_object
      borrowing object
analysis_object
      analysis object
```

---

UniformPrior-class      UniformPrior *Class*

---

### Description

A class for defining uniform priors to be translated to Stan code. Objects of class UniformPrior should not be created directly but by the constructor `uniform_prior()`.

### Slots

`stan_code` character. Stan implementation of the prior, with placeholders for uniform stan function parameters surrounded with `{ { and } }` to be replaced with `glue::glue()`.

`n_param` integer. Number of prior parameters (2).

`constraint` character. Support of prior distribution, "`<lower=alpha, upper=beta>`".

`alpha` numeric. Lower bound.

`beta` numeric. Upper bound (`>alpha`).

### See Also

Other prior classes: [Prior-class](#), [PriorBernoulli-class](#), [PriorBeta-class](#), [PriorCauchy-class](#), [PriorExponential-class](#), [PriorGamma-class](#), [PriorHalfCauchy-class](#), [PriorHalfNormal-class](#), [PriorNormal-class](#), [PriorPoisson-class](#)

---

uniform\_prior      *Prior uniform distribution*

---

### Description

Prior uniform distribution

### Usage

```
uniform_prior(alpha, beta)
```

### Arguments

`alpha`                  numeric. Lower bound.

`beta`                    numeric. Upper bound (`>alpha`).

### Details

Stan reference <https://mc-stan.org/docs/functions-reference/uniform-distribution.html>

**Value**

Object of class `UniformPrior`.

**See Also**

Other priors: `prior_bernoulli()`, `prior_beta()`, `prior_cauchy()`, `prior_exponential()`, `prior_gamma()`, `prior_half_cauchy()`, `prior_half_normal()`, `prior_normal()`, `prior_poisson()`

**Examples**

```
up <- uniform_prior(0, 1)
```

---

variable\_dictionary    *Create Variable Dictionary*

---

**Description**

Create Variable Dictionary

**Usage**

```
variable_dictionary(analysis_obj)
```

**Arguments**

analysis\_obj    Analysis. Object to describe variable names.

**Value**

A data.frame with the names of Stan variables and the descriptions.

---

weib\_ph\_surv\_dist    *Legacy function for the Weibull proportional Hazards survival distribution*

---

**Description**

Please use `outcome_surv_weibull_ph()` instead.

**Usage**

```
weib_ph_surv_dist(...)
```

**Arguments**

...    Deprecated arguments to `weib_ph_surv_dist()`.

**Value**

This function does not return a value. When called, it triggers an error message indicating that `weib_ph_surv_dist()` is deprecated and that `outcome_surv_weibull_ph()` should be used instead.

# Index

- \* **borrowing classes**
  - Borrowing-class, 13
  - BorrowingFixedPowerPrior-class, 14
  - BorrowingFull-class, 14
  - BorrowingHierarchicalCommensurate-class, 15
  - BorrowingNone-class, 15
- \* **borrowing**
  - borrowing\_full, 17
  - borrowing\_none, 19
- \* **datasets**
  - example\_matrix, 40
  - example\_surv, 40
- \* **outcome models**
  - outcome\_bin\_logistic, 59
  - outcome\_cont\_normal, 60
  - outcome\_surv\_exponential, 61
  - outcome\_surv\_pem, 63
  - outcome\_surv\_weibull\_ph, 64
- \* **outcome**
  - BinaryOutcome-class, 11
  - ContinuousOutcome-class, 24
  - Outcome-class, 56
  - OutcomeBinaryLogistic-class, 56
  - OutcomeContinuousNormal-class, 57
  - OutcomeSurvExponential-class, 57
  - OutcomeSurvPEM-class, 58
  - OutcomeSurvWeibullPH-class, 59
  - TimeToEvent-class, 98
- \* **prior classes**
  - Prior-class, 69
  - PriorBernoulli-class, 70
  - PriorBeta-class, 70
  - PriorCauchy-class, 71
  - PriorExponential-class, 71
  - PriorGamma-class, 72
  - PriorHalfCauchy-class, 72
  - PriorHalfNormal-class, 73
  - PriorNormal-class, 73
  - PriorPoisson-class, 74
  - UniformPrior-class, 102
- \* **priors**
  - prior\_bernoulli, 74
  - prior\_beta, 75
  - prior\_cauchy, 76
  - prior\_exponential, 76
  - prior\_gamma, 77
  - prior\_half\_cauchy, 78
  - prior\_half\_normal, 78
  - prior\_normal, 79
  - prior\_poisson, 80
  - uniform\_prior, 102
- \* **simulation classes**
  - sim\_borrowing\_list, 91
  - sim\_covariate\_list, 93
  - sim\_data\_list, 94
  - sim\_outcome\_list, 96
  - sim\_treatment\_list, 98
- \* **simulation**
  - sim\_covariates, 92
  - sim\_samplesize, 97
- \* **simvar classes**
  - SimVarBin-class, 89
  - SimVarCont-class, 90
- \* **simvar**
  - bin\_var, 12
  - cont\_var, 24
- .analysis\_obj (Analysis-class), 6
- .baseline\_data\_list
  - (BaselineDataList-class), 8
- .baseline\_dataframe
  - (BaselineDataFrame-class), 8
- .baseline\_object
  - (BaselineObject-class), 9
- .bin\_var (SimVarBin-class), 89
- .borrowing\_fixed\_power\_prior
  - (BorrowingFixedPowerPrior-class), 14

- .borrowing\_full (BorrowingFull-class), 14
- .borrowing\_hierarchical\_commensurate (BorrowingHierarchicalCommensurate-class), 15
- .borrowing\_none (BorrowingNone-class), 15
- .cont\_var (SimVarCont-class), 90
- .covariate\_class (Covariates-class), 26
- .datasim\_cut\_off (DataSimCutOff-class), 37
- .datasim\_enrollment (DataSimEnrollment-class), 37
- .datasim\_event (DataSimEvent-class), 37
- .datasim\_fixed\_external\_data (DataSimFixedExternalData-class), 38
- .datasim\_object (DataSimObject-class), 38
- .mcmc\_simulation\_result (MCMCSimulationResult-class), 52
- .outcome\_bin\_logistic (OutcomeBinaryLogistic-class), 56
- .outcome\_cont\_normal (OutcomeContinuousNormal-class), 57
- .outcome\_surv\_exponential (OutcomeSurvExponential-class), 57
- .outcome\_surv\_pem (OutcomeSurvPEM-class), 58
- .outcome\_surv\_weibull\_ph (OutcomeSurvWeibullPH-class), 59
- .prior\_bernoulli (PriorBernoulli-class), 70
- .prior\_beta (PriorBeta-class), 70
- .prior\_cauchy (PriorCauchy-class), 71
- .prior\_exponential (PriorExponential-class), 71
- .prior\_gamma (PriorGamma-class), 72
- .prior\_half\_cauchy (PriorHalfCauchy-class), 72
- .prior\_half\_normal (PriorHalfNormal-class), 73
- .prior\_normal (PriorNormal-class), 73
- .prior\_poisson (PriorPoisson-class), 74
- .sim\_borrowing\_list (SimBorrowingList-class), 86
- .sim\_covariate\_list (SimCovariateList-class), 86
- .sim\_covariates (SimCovariates-class), 87
- .sim\_data\_list (SimDataList-class), 87
- .sim\_outcome\_list (SimOutcomeList-class), 88
- .sim\_sample\_size (SimSampleSize-class), 88
- .sim\_treatment\_list (SimTreatmentList-class), 88
- .simulation\_obj (Simulation-class), 89
- .treatment\_class (Treatment-class), 99
- .uniform\_prior (UniformPrior-class), 102
- add\_covariates, 6
- add\_covariates(), 26, 27, 30
- Analysis, 27
- Analysis-class, 6
- as.data.frame.BaselineDataList (as\_data\_frame), 7
- as\_data\_frame, 7
- baseline\_covariates, 9
- baseline\_covariates(), 9, 28
- BaselineDataFrame, 43, 85
- BaselineDataFrame-class, 8
- BaselineDataList, 7
- BaselineDataList-class, 8
- BaselineObject, 10, 29
- BaselineObject-class, 9
- bernoulli\_prior, 10
- beta\_prior, 10
- bin\_var, 12, 25
- binary\_cutoff, 11
- binary\_cutoff(), 28, 85
- BinaryOutcome-class, 11
- Borrowing, 27
- Borrowing-class, 13
- borrowing\_details, 16
- borrowing\_fixed\_power\_prior, 16
- borrowing\_full, 17, 19
- borrowing\_full(), 13, 14, 27
- borrowing\_hierarchical\_commensurate, 18



- get\_vars, BorrowingFixedPowerPrior-method (get\_vars), 48
- get\_vars, ContinuousOutcome-method (get\_vars), 48
- get\_vars, Covariates-method (get\_vars), 48
- get\_vars, NULL-method (get\_vars), 48
- get\_vars, SimBorrowingList-method (get\_vars), 48
- get\_vars, SimCovariateList-method (get\_vars), 48
- get\_vars, SimOutcomeList-method (get\_vars), 48
- get\_vars, SimTreatmentList-method (get\_vars), 48
- get\_vars, Simulation-method (get\_vars), 48
- get\_vars, TimeToEvent-method (get\_vars), 48
- get\_vars, Treatment-method (get\_vars), 48
- glue::glue(), 69–74, 102
- half\_cauchy\_prior, 49
- half\_normal\_prior, 50
- load\_and\_interpolate\_stan\_model, 50
- load\_stan\_file, 51
- logistic\_bin\_outcome, 51
- mcmc\_sample, 52
- mcmc\_sample, Analysis-method (mcmc\_sample), 52
- mcmc\_sample, ANY-method (mcmc\_sample), 52
- mcmc\_sample, Simulation-method (mcmc\_sample), 52
- MCMCSimulationResult-class, 52
- normal\_prior, 55
- null\_event\_dist (create\_event\_dist), 31
- Outcome, 27
- Outcome-class, 56
- outcome\_bin\_logistic, 59, 61, 62, 64, 65
- outcome\_bin\_logistic(), 27, 56
- outcome\_cont\_normal, 60, 60, 62, 64, 65
- outcome\_cont\_normal(), 57
- outcome\_surv\_exponential, 60, 61, 61, 64, 65
- outcome\_surv\_exponential(), 27, 57
- outcome\_surv\_pem, 60–62, 63, 65
- outcome\_surv\_pem(), 58
- outcome\_surv\_weibull\_ph, 60–62, 64, 64
- outcome\_surv\_weibull\_ph(), 27, 59
- OutcomeBinaryLogistic, 60
- OutcomeBinaryLogistic-class, 56
- OutcomeContinuousNormal, 61
- OutcomeContinuousNormal-class, 57
- OutcomeSurvExponential, 62
- OutcomeSurvExponential-class, 57
- OutcomeSurvPEM, 64
- OutcomeSurvPEM-class, 58
- OutcomeSurvWeibullPH, 65
- OutcomeSurvWeibullPH-class, 59
- par(), 66
- plot, 65
- plot(), 67, 68
- plot, Prior, missing-method (plot), 65
- plot, PriorBernoulli, missing-method (plot), 65
- plot, PriorBeta, missing-method (plot), 65
- plot, PriorCauchy, missing-method (plot), 65
- plot, PriorExponential, missing-method (plot), 65
- plot, PriorGamma, missing-method (plot), 65
- plot, PriorHalfCauchy, missing-method (plot), 65
- plot, PriorHalfNormal, missing-method (plot), 65
- plot, PriorNormal, missing-method (plot), 65
- plot, PriorPoisson, missing-method (plot), 65
- plot, UniformPrior, missing-method (plot), 65
- plot\_pdf, 67
- plot\_pmf, 68
- poisson\_prior, 68
- possible\_data\_sim\_vars, 69
- posterior::draws, 81
- Prior-class, 69
- prior\_bernoulli, 74, 75–80, 103
- prior\_bernoulli(), 69, 70
- prior\_beta, 75, 75, 76–80, 103
- prior\_beta(), 69, 70
- prior\_cauchy, 75, 76, 77–80, 103

- prior\_cauchy(), 69, 71
- prior\_exponential, 75, 76, 76, 77–80, 103
- prior\_exponential(), 71
- prior\_gamma, 75–77, 77, 78–80, 103
- prior\_gamma(), 69, 72
- prior\_half\_cauchy, 75–77, 78, 79, 80, 103
- prior\_half\_cauchy(), 69, 72
- prior\_half\_normal, 75–78, 78, 80, 103
- prior\_half\_normal(), 73
- prior\_normal, 75–79, 79, 80, 103
- prior\_normal(), 69, 73
- prior\_poisson, 75–80, 80, 103
- prior\_poisson(), 69, 74
- PriorBernoulli, 74
- PriorBernoulli-class, 70
- PriorBeta, 75
- PriorBeta-class, 70
- PriorCauchy, 76
- PriorCauchy-class, 71
- PriorExponential, 77
- PriorExponential-class, 71
- PriorGamma, 77
- PriorGamma-class, 72
- PriorHalfCauchy, 78
- PriorHalfCauchy-class, 72
- PriorHalfNormal, 79
- PriorHalfNormal-class, 73
- PriorNormal, 79
- PriorNormal-class, 73
- PriorPoisson, 80
- PriorPoisson-class, 74
- rect(), 68
- rename\_draws\_covariates, 81
- set\_cut\_off, 82
- set\_dropout, 82
- set\_enrollment, 83
- set\_transformations, 84
- set\_transformations, BaselineObject-method, 85
- show\_guide, 86
- show\_guide, Simulation-method (show\_guide), 86
- sim\_borrowing\_list, 91, 94, 95, 97, 98
- sim\_covariate\_list, 91, 93, 95, 97, 98
- sim\_covariates, 92, 97
- sim\_covariates\_summ, 93
- sim\_data\_list, 91, 94, 94, 97, 98
- sim\_outcome\_list, 91, 94, 95, 96, 98
- sim\_samplesize, 92, 97
- sim\_treatment\_list, 91, 94, 95, 97, 98
- SimBorrowingList, 91
- SimBorrowingList-class, 86
- SimCovariateList, 94
- SimCovariateList-class, 86
- SimCovariates-class, 87
- SimDataList, 21, 43, 44, 95
- SimDataList-class, 87
- SimOutcomeList, 97
- SimOutcomeList-class, 88
- SimSampleSize-class, 88
- simSurv::simSurv, 31
- simSurv::simSurv(), 37
- SimTreatmentList, 98
- SimTreatmentList-class, 88
- Simulation, 33
- Simulation-class, 89
- SimVar-class, 89
- SimVarBin, 13
- SimVarBin-class, 89
- SimVarCont, 24
- SimVarCont-class, 90
- TimeToEvent-class, 98
- Treatment, 27, 100
- Treatment-class, 99
- treatment\_details, 100
- treatment\_details(), 27, 99
- trim\_cols, 100
- trim\_cols, Borrowing-method (trim\_cols), 100
- trim\_cols, BorrowingFixedPowerPrior-method (trim\_cols), 100
- trim\_cols, BorrowingHierarchicalCommensurate-method (trim\_cols), 100
- trim\_rows, 101
- trim\_rows, Borrowing-method (trim\_rows), 101
- trim\_rows, BorrowingNone-method (trim\_rows), 101
- uniform\_prior, 75–80, 102
- uniform\_prior(), 69, 102
- UniformPrior, 103
- UniformPrior-class, 102
- variable\_dictionary, 103

`weib_ph_surv_dist`, [103](#)